CALIDAD DE LOS PROYECTOS DE SOFTWARE: REVISIONES UTILIZANDO RAZONAMIENTO BASADO EN CASOS

Resumen / *Abstract*

En este trabajo se expone un sistema que permite planificar, controlar y dar seguimiento a las inspecciones realizadas a los proyectos de software, que utiliza el razonamiento basado en casos para planificar las inspecciones. Además, se presentan los conceptos fundamentales relacionados con las inspecciones dentro del sistema de aseguramiento de calidad, así como el procedimiento para llevarlas a cabo. También se expone una breve panorámica del razonamiento basado en casos.

In this work a system is exposed that allows to plan, to control and to give pursuit to the inspections carried out to the software projects that it uses the based reasoning in cases to plan the inspections. The fundamental concepts related with the inspections are also presented inside the system of insurance of quality, as well as the procedure to carry out them. A brief panoramic of the based reasoning is also given in cases.

Palabras clave / *Key words*

Aseguramiento de calidad de software, razonamiento basado en casos, ingeniería de software, inteligencia artificial

Software quality assurance, case-based reasoning, software engineering, artificial intelligent

INTRODUCCIÓN

Los conceptos generales de calidad han evolucionado y se han vuelto más sofisticados. En 1970, se centraron en el producto y sugirieron que los productos y servicios tuviesen claras especificaciones como tamaño, peso, color, duración de la batería o tiempo entre fallas. La idea es que los consumidores pudieran esperar un producto y un desempeño rentables.

Pero la visión de calidad en los productos de software tiene otra dimensión. Los usuarios de software esperan una continua progresión de las características: la promesa de actualización; alto desempeño y confiabilidad; facilidad de instalación; utilización y mantenimiento.

En ingeniería como en administración se entiende por calidad: "El conjunto de atributos que poseen los bienes o servicios para satisfacer los requerimientos de los consumidores".[1]

La calidad, entonces, es un conjunto de atributos que poseen tanto productos como servicios y que hace que los consumidores inclinen sus preferencias hacia algunos de ellos. Se asume que es claro que la calidad no surge de manera espontánea y que para obtenerla se requiere realizar esfuerzos. Así pues, la preocupación de los desarrolladores de software es cómo lograr la calidad al elaborar sus productos.

Se considera entonces que la calidad del software es:

"El grado en el que el software satisface una serie de requerimientos de operación preestablecidos, los estándares de desarrollo especificados con anterioridad y las características inherentes a todo producto de software desarrollado de manera profesional".[2]

***Martha D. Delgado Dapena**, Ingeniera Informática, Instructora, Centro de Estudios de Ingeniería de Sistemas (CEIS), Instituto Superior Politécnico José Antonio Echeverría, Cujae, Ciudad de La Habana, Cuba*
e-mail:marta@ceis.cujae.edu.cu

Recibido: Noviembre del 2002
Aprobado: Enero del 2003

Varias son las definiciones de calidad dadas por los especialistas, pero todos parecen estar de acuerdo con los criterios siguientes:[1-3]

1. En la calidad del software influyen múltiples factores, entre los que se encuentran el establecimiento de métodos que se utilizan en el proceso de desarrollo y el desarrollo del propio proceso de elaboración del software.

2. Un software con calidad es aquel que cumple con los requerimientos de los clientes y que tiene ausencia de defectos.

3. Un software con calidad debe cumplir los siguientes criterios: confiabilidad, eficiencia, integridad, facilidad de mantenimiento, flexibilidad, facilidad de prueba, entre otros.[4]

4. Establecer un sistema de aseguramiento de la calidad en las empresas de software contribuye notablemente a elevar la calidad del producto final.[5]

Esencialmente, el aseguramiento de la calidad del software consiste en la revisión de los productos y su documentación relacionada, para verificar su cobertura, corrección, confiabilidad y facilidad de mantenimiento. Y, por supuesto, incluye la garantía de que un sistema cumpla las especificaciones y los requerimientos para el uso y desempeño deseados. En la figura 1 se muestra un gráfico presentado a partir de un estudio realizado en empresas productoras de software,[6] que incluye una comparación entre sistemas que han sido construidos con alta calidad y sistemas construidos con escasa calidad, lo que muestra que en los sistemas desarrollados con alta calidad se disminuyen considerablemente los costos en el mantenimiento, pues estos sistemas han sido sometidos a revisiones que detectan y eliminan defectos en etapas tempranas de su construcción.

Dentro del sistema de aseguramiento de calidad las inspecciones o revisiones en las diferentes etapas de desarrollo del software desempeña un importante papel, pues ellas contribuyen a detectar defectos en las etapas tempranas y por tanto a mejorar la calidad del producto.

Tradicionalmente las inspecciones se realizan al finalizar el software y esto trae consigo los siguientes problemas:

- La inspección final es inoperante, pues no mejora la calidad de un producto, solo descubre qué productos son buenos y cuáles no.
- La inspección final es costosa, pues implica:
  - Cubrir gastos de un departamento que en realidad no produce.
  - Asumir los gastos de los productos que hay que rehacer y de los que es necesario desechar.
- La inspección final afecta la moral de los trabajadores, pues hace aparecer que el producto defectuoso es culpa de ellos y no del sistema.
- El hecho de hacer la inspección final implica que la administración acepta:
  - Trabajar con un proceso mal planeado.
  - Contar siempre con un porcentaje de productos defectuosos

La calidad del software puede medirse después de elaborado el producto, pero esto puede resultar muy costoso si se detectan problemas derivados de imperfecciones en el diseño, por lo que es imprescindible tener en cuenta tanto la obtención de la calidad como su control durante todas las etapas del ciclo de vida de software.[3,7] Más que recurrir a una inspección final, se debe atender al proceso mismo, detectando los defectos y poniendo las acciones correctivas correspondientes para prevenirlos en lo adelante.

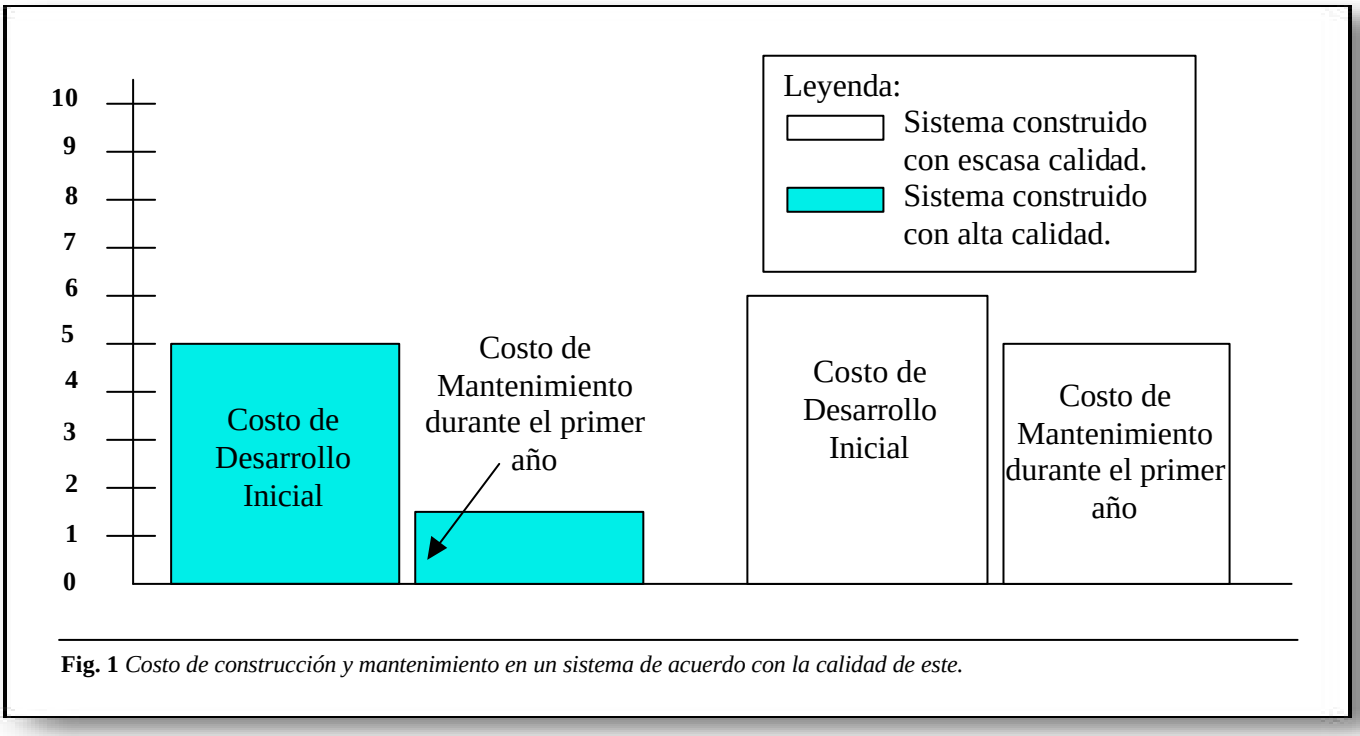

Fig. 1 *Costo de construcción y mantenimiento en un sistema de acuerdo con la calidad de este.*

En este trabajo se expone la situación de la empresa cubana de software y la importancia de introducir las **revisiones** como parte del Sistema de Aseguramiento de Calidad del Software. Además, se propone un procedimiento para aplicar las inspecciones, así como los momentos en que sería más adecuado ejecutar tales **revisiones** y se hace una propuesta de una lista de chequeo para la inspección a la etapa de estudio preliminar.

DESARROLLO DE SOFTWARE EN CUBA

El desarrollo de una Industria Nacional de Software es una área de gran prioridad para el Estado cubano debido a la alta perspectiva económica que posee, así como para el aseguramiento de un grupo importante de actividades del país. A pesar de ello, los resultados alcanzados no cubren las expectativas, ya que la productividad es baja, la cantidad real de recursos a consumir -en tiempo principalmente- es casi impredecible y el trabajo realizado casi nunca tiene la calidad y profesionalidad requerida. Los proyectos están excesivamente tarde y los beneficios que pudieran obtenerse al utilizar los mejores métodos e instrumentos en las distintas etapas no se detectan en este medio indisciplinado y caótico de desarrollo.[8,9]

En estudios realizados por el Centro de Estudios de Ingeniería de Sistemas (CEIS) en empresas nacionales se detectaron problemas relacionados con:[10]

- El personal disponible en estas empresas es aún escaso aunque tiene un alto nivel de preparación.
- Existe una gran desorganización en las empresas lo que no permite aplicar técnicas, modelos o estándares que ayudarían al desarrollo de esta.
- Existen pocos clientes y no se ha creado una disciplina para el intercambio con ellos.
- Resulta muy pobre el mercado externo.
- No existe una cultura de producción de software bajo parámetros de terminación y calidad, donde se actúe bajo conceptos y estudios técnicamente fundamentados por equipos multidisciplinarios y competentes dirigidos a la creación de un producto orientado a determinado mercado.
- Hay mala calidad en gran parte del software que se produce en el país y es indispensable la solución de este problema lo más breve posible, pues la calidad del producto que desarrollan las empresas nacionales es clave para mejorar su competitividad, y teniendo la calidad del producto como elemento distintivo, estas pueden encontrar nuevos mercados.

Los resultados de este estudio arrojan la necesidad de establecer políticas de calidad en las organizaciones del país con el fin de mejorar la productividad de las empresas y lograr una adecuada satisfacción de los clientes tanto en el mercado nacional como en el internacional.

A continuación se detallan las características del proceso de inspección y algunas propuestas para introducirlo en empresas cubanas desarrolladoras de software, aunque pudiera ser extendido también a otras empresas con características similares.

PROGRAMACIÓN DE INSPECCIONES

El proceso de inspección tiene gran importancia en el Sistema de Aseguramiento de Calidad, pues este ayuda a detectar defectos en etapas tempranas del desarrollo de los proyectos de software, lo que mejora considerablemente la calidad del producto final. Algo esencial en cualquier proyecto de desarrollo de software es la definición e implementación de las actividades necesarias para valorar y medir la calidad del producto de software producido por cada proyecto, de acuerdo con los requerimientos establecidos para este.

Entre los objetivos de las inspecciones están:[4,11]

- Descubrir errores de funcionamiento, lógica o de implementación.
- Verificar que el software cumpla con los requisitos.
- Garantizar que se cumplan los estándares establecidos.
- Conseguir un desarrollo uniforme.
- Lograr que los proyectos sean manejables.

Las inspecciones pudieran realizarse en cualquier fase del desarrollo del proyecto, decisión que debe ser tomada por el planificador del proyecto de software conjuntamente con el jefe de proyecto y el administrador de aseguramiento de calidad y documentada debidamente en el Plan de Revisión y Auditoría del Proyecto de Software. No obstante, hay varios momentos en los que no deberían omitirse; el primero al finalizar la etapa de estudio preliminar y definición de los requerimientos del software, para detectar lo antes posible defectos relacionados con los requerimientos contratados, las necesidades de los usuarios e involucrados en el proyecto y otros que pueden dar al traste con un software que no represente adecuadamente las expectativas de los clientes. Otras inspecciones deben realizarse al concluir la etapas de diseño y codificación respectivamente, para garantizar la eliminación de la mayor cantidad posible de defectos antes de pasar a la etapa siguiente.

La última inspección debe llevarse a cabo, una vez terminada la primera versión del software, de manera tal que se verifique si efectivamente el producto terminado posee todos los requisitos técnicos y funcionales necesarios que le permitan cumplir con los objetivos y requerimientos contratados y satisfacer adecuadamente las necesidades de los clientes.

PROCEDIMIENTO PARA DESARROLLAR LAS REVISIONES O INSPECCIONES

Una vez llegado el momento de la inspección, definido en el Plan de Revisión y Auditoria del Proyecto de Software,[5] el procedimiento a seguir para desarrollar la inspecciones en la empresa de software debe ser el siguiente:

1. Planificación

El moderador designado para tal efecto deberá establecer la conducta y progreso de la misma para lo que llevará a cabo las tareas ralacionadas a continuación:

- Identificación del equipo de inspección, asegurándose de que sus miembros estén disponibles para preparar adecuadamente la inspección.
- Verificación de que los materiales necesarios estén disponibles y de acuerdo con los estándares.

• Verificación de que están establecidos los criterios de entrada para la inspección, en forma de lista de chequeo.

• Programar tiempo y lugar de la reunión de inspección, asegurando que el lugar esté reservado para tales fines.

• Asignar la función de lector a un miembro seleccionado del equipo de inspección.

• Citar a todos los miembros del equipo y otras partes interesadas para la reunión de inspección.

2. Revisión.

• El moderador programará un contacto con los miembros del equipo de inspección y el autor del producto a inspeccionar, donde este último dará una breve descripción del software, qué tareas se han realizado, cómo se realizará cada tarea, una descripción de las interfaces y otros aspectos que pudieran resultar de interés y que serán acordados previamente entre el autor y el moderador. Si existiese algún problema significativo que viole los criterios de entrada establecidos, la inspección puede ser pospuesta hasta que el problema sea solucionado.

• El moderador citará a todos para la reunión de inspección y entregará los materiales a los miembros del equipo de inspección.

3. Preparación.

• Los inspectores revisan el producto y documentan cualquier discrepancia o defecto encontrado. Deben mediar como mínimo cinco días, por supuesto, este tiempo o fase de preparación dependerá de la complejidad del producto a inspeccionar y será determinado por el moderador.

• Durante esta fase los inspectores registran los defectos encontrados en un documento para tales efectos y que será suministrado, conjuntamente con los materiales del producto a inspeccionar, por el moderador.

• El lector debe registrar cualquier dificultad en el entendimiento de los materiales facilitados a los inspectores.

4. Reunión de inspección.

• El moderador es responsable de conducir adecuadamente y asegurar que los miembros del equipo trabajen con profesionalidad, encontrando defectos en el producto que contribuyan a mejorar su calidad.

• El lector deberá mostrar el producto de forma lógica para que se vayan exponiendo las discrepancias encontradas.

• Los defectos son identificados, discutidos y registrados, atención particular merecen aquellos que no han sido detectados con anterioridad y que surgen como consecuencia de la discusión de los miembros del equipo.

• El objetivo de la reunión es encontrar defectos, no soluciones, esto último es responsabilidad del autor.

• Como resultado de la reunión se deberá elaborar un documento que contenga los defectos encontrados y que será entregado al autor y Grupo de Aseguramiento de la Calidad de Software.

5. Reproceso.

• El autor comienza a corregir los defectos y cuando estén listos lo informará al moderador.

6. Verificación

• La corrección de los defectos es verificada.

• Si el moderador entiende que se requiere una reinspección, lo informará al autor y este comenzará a preparar el material necesario.

• Si es necesario el moderador puede designar a un miembro del equipo para que lo asista en la revisión de los defectos y una vez verificado, cada uno debe ser registrado en el documento correspondiente.

REVISIONES UTILIZANDO RAZONAMIENTO BASADO EN CASOS

Cada empresa puede definir sus propias políticas y estándares y en ellos definirá, entre otros, qué notaciones y herramientas serán empleadas por los grupos de desarrollo para describir las especificaciones de los proyectos, así como la estrategia a seguir en la planificación de las inspecciones. En este caso sería conveniente que la empresa dispusiera de una herramienta que le permitiera inspeccionar teniendo en cuenta las políticas definidas y además la experiencia de inspecciones anteriores realizadas en la propia empresa.

Una buena alternativa es contar con una herramienta que partiendo del tipo de proyecto proponga en qué momentos deben realizarse las inspecciones, qué tipo de inspecciones y para cada una de ellas la lista de chequeo más conveniente de acuerdo con la experiencia acumulada por la empresa en inspecciones realizadas a proyectos con características similares.

CONCEPTOS FUNDAMENTALES DE RAZONAMIENTO BASADO EN CASOS

El RBC es una técnica de inteligencia artificial que se basa en almacenar ejemplos a partir de los cuales se pueden proponer soluciones a nuevos problemas, utilizando la experiencia acumulada hasta el momento.[12] Esta técnica intenta llegar a la solución de nuevos problemas, de forma similar a como lo hacen los seres humanos.[13]

El funcionamiento del RBC parte de estos principios y para ello comprende cuatro actividades principales:[14,15]

- Recuperar los casos más parecidos.
- Reutilizar el o los casos para tratar de resolver el nuevo problema.
- Revisar y adaptar la solución propuesta, en caso de ser necesario.
- Almacenar la nueva solución como parte de un nuevo caso.

Un nuevo problema se compara con los casos almacenados previamente en la base de casos y se recuperan uno o varios casos. Posteriormente se utiliza y evalúa una solución, sugerida por los casos que han sido seleccionados con anterioridad, para ver si se aplica al problema actual. A menos que el caso recuperado sea igual al actual, la solución probablemente tendrá que ser revisada y adaptada, produciéndose un nuevo caso que será almacenado.

La forma de representar y almacenar estas experiencias se realiza a través de casos. Un caso mantiene todos los atributos y características relevantes de un evento pasado. Estas características servirán como índices para la recuperación de caso futuro.

Para la recuperación de los casos existen varias técnicas, entre las que se encuentra el método del vecino más próximo.[12] El caso más parecido será aquel que más próximo esté del problema actual en cuanto a sus atributos, es decir, será el caso cuya evaluación de la función objetivo tome el mejor valor dentro de los casos contemplados en la base de casos existente, entendiendo como

valor más próximo el valor de la función objetivo que denote menor diferencia entre el caso actual y el caso evaluado.

El RBC ha empezado a desepeñar un papel importante en la ingeniería de software y del conocimiento, en la informática comercial y en la gestión del conocimiento.[14,16] Se observa un rápido incremento en el número de aplicaciones disponibles, especialmente de soporte técnico, atención al cliente, comercio electrónico y gestión del conocimiento corporativo. La utilización de esta técnica en áreas de la ingeniería de software puede contribuir notablemente a elevar la calidad del proceso de desarrollo de software.

SISTEMA DE CONTROL Y SEGUIMIENTO DE INSPECCIONES A PROYECTOS DE SOFTWARE UTILIZANDO RBC

En el Centro de Estudios de Ingeniería de Sistemas (CEIS) se ha desarrollado un sistema que permite hacer un seguimiento de las inspecciones a los proyectos de la empresa, desde el momento en que se define el plan de aseguramiento de la calidad, que incluye las inspecciones, pasando por la definición de los miembros del equipo de inspectores, incluyendo las posibles listas de chequeo y terminando con la documentación final de la inspección que se almacena para que pueda ser consultada y así comparar los resultados que produce la revisión en la calidad final de producto de software. Además, permitirá llevar métricas asociadas a los defectos, lo que proporciona a la empresa una base de datos con elementos a tener en cuenta en el momento de analizar los procesos involucrados en la producción del software.

La aplicación utiliza el RBC[15] para proponer el plan de revisiones, con la lista de chequeo asociada a cada inspección. Estas propuestas solo constituyen un punto de partida, pues el administrador de aseguramiento de la calidad podrá enriquecer el plan y las listas de chequeo (tabla 1) para cada proyecto si lo entiende conveniente. Esta herramienta permite aprovechar la experiencia acumulada por los inspectores y tener en cuenta las políticas definidas en la empresa donde se desarrolla el producto que se desea inspeccionar.

Está claro que la lista de chequeo propuesta en la tabla 1 es válida si se utilizan ciertas notaciones para describir las especificaciones del proyecto,[17,18] por ejemplo, para que se pueda inspeccionar el diagrama de casos de uso del proyecto es necesario que se haya utilizado notación UML, [18,20] para describir dichas especificaciones, en cualquier otro caso habría que reajustar la lista de chequeo.

Los parámetros de entrada a la opción de propuesta de planificación son, en primer lugar, el tipo de proyecto y las listas de estándares o políticas definidas por la empresa para el desarrollo del proyecto a inspeccionar, esta información será suministrada por el administrador de aseguramiento de la calidad. Además, es necesario que para planificar el sistema conozca el resultado de la efectividad y otras métricas asociadas a inspecciones realizadas con anterioridad a otros proyectos del mismo tipo, información que se obtiene a partir de la información almacenada en la Base de Casos.

El sistema cuenta con una base de datos que contiene la información detallada de las inspecciones realizadas a los proyectos. A continuación se relacionan las entidades fundamentales:

- Proyecto.
  - Tipo de proyecto: aplicación web, inteligencia artificial, sistema en tiempo real, sistema de gestión tradicional etcétera.
  - Lista de los estándares y políticas de la empresa que utiliza este proyecto.
  - Detalle de las características del proyecto (título, avance, documentación, etcétera).
  - Listado de inspecciones al proyecto.
- Inspección realizada a un proyecto.
  - Tipo de inspección.
  - Tipo de proyecto.
  - Momento en que se realiza (etapa).
  - Lista de chequeo.
  - Resultados de las métricas.

La base de casos[14] que será enriquecida con la experiencia de la propia empresa contiene la siguiente información para cada caso:

- Tipo de proyecto.
- Lista de los estándares y políticas de la empresa que utiliza este proyecto.
- Para cada inspección:
  - Tipo de inspección.
  - Momento del proyecto en que se realiza la inspección (etapa).
  - Lista de chequeo.
  - Resultados de las métricas asociadas.

El usuario del sistema deberá suministrar el tipo de proyecto que quiere inspeccionar y cuáles de las políticas definidas por la empresa, él utiliza, por ejemplo puede ser un sistema de gestión tradicional, para el control de las citas de los pacientes en un hospital y la notación utilizada para describir las especificaciones del proyecto puede ser UML, [11,21] que es una de las definidas por la empresa. Con esta información la herramienta selecciona los casos almacenados que se correspondan con los parámetros suministrados y decide cuál o cuáles de ellos deben ser considerados en la solución que se le propone al usuario. Una vez definido el conjunto de casos involucrados se elabora una propuesta de plan de inspección valorando en cada caso el resultado de las métricas.

Para la recuperación de los casos se utilizarán índices por tipo de proyecto, esto permitirá localizar con rapidez cuáles de los casos almacenados pueden ser considerados en la solución del nuevo problema presentado. Para determinar los casos que definitivamente van a ser considerados para construir la nueva solución se utiliza la siguiente función de similitud:[15]

$$\text{Similitud (O, A)} = \sqrt{(P_1 * CPI)^2 + (P_2 * CPSI)^2}$$

donde:

- CPI: Cantidad de políticas iguales entre el caso actual y el almacenado.
- CPSI: Cantidad de políticas imprescindibles, del caso a planificar, que están presentes en el caso almacenado.
- P_1 y P_2 : Pesos que tendrán asignados los atributos CPI y CPSI respectivamente, y que serán decididos por el administrador de proyectos cada vez que se introduzca un nuevo tipo de proyecto.

Tabla 1
Lista de chequeo propuesta por el sistema para inspeccionar un proyecto al finalizar la etapa de Estudio Preliminar

- La documentación del proyecto se corresponde con los estándares definidos por la empresa (metodologías y otros).
- Están consideradas todas las necesidades de los clientes en los requerimientos del sistema.
- Están documentados todos los requerimientos contratados.
- Correcta evaluación y selección de las alternativas de solución.
- Están documentados todos los requisitos no funcionales.
- Están documentados todos los requisitos adicionales.
- Existe glosario de términos.
- Todos los objetivos del proyecto están reflejados en los requerimientos.
- Correspondencia de los requerimientos con el planteamiento del problema.
- Para cada requerimiento funcional:
  - Definición y descripción de sus atributos.
  - Se corresponde con alguna necesidad de usuario/involucrado.
  - Documentación de los riesgos con su prioridad.
  - Clasificación de acuerdo con los riesgos.
  - Está reflejado en algún caso de uso.
- Se han detectado todos los actores.
- Para cada actor:
  - Documentar su caracterización, que debe incluir al menos sus necesidades y responsabilidades.
  - Se relaciona con al menos un caso de uso.
- Para cada caso de uso:
- Contiene descripción, actores, etcétera.
  - Responde a algún requerimiento funcional.
- La planificación de los ciclos en que se desarrollará se corresponde con los riesgos de los requerimientos asociados.
  - Ofrece algún resultado de valor para al menos un actor.
  - Está correctamente nombrado.
  - En el diagrama de casos de uso se relaciona con al menos un actor.
  - Considera los requisitos no funcionales asociados.
  - Considera los requisitos adicionales asociados.
  - Están descritas las interfaces-usuario para cada relación actor-caso de uso.
- Para cada interfaz-usuario:
  - Permite que el actor navegue de forma adecuada.
  - Proporciona una apariencia agradable.
  - Utiliza el lenguaje propio del negocio.
  - Proporciona una forma consistente de trabajo con la interfaz usuario.
  - Cumple con estándares relevantes como: color, señalizaciones, barras de herramientas, etcétera.
- Uniformidad entre las interfaces usuario del prototipo, considerando entre otras, color, tamaño, barras de herramientas y tamaños de componentes.
- En el diagrama de casos de uso:
  - No existen actores desconectados.
  - Todos los actores han sido descritos.
  - No existen casos de uso desconectados.
  - Todos los casos de uso han sido descritos.
- Se realizó el estudio de factibilidad adecuadamente.
- Se realizó la estimación del proyecto adecuadamente.
- Se planificó el proyecto adecuadamente.
- Están documentados los riesgos.
- Para cada riesgo:
  - Descripción.
  - Prioridad.
  - Impacto.
  - Monitor.
  - Responsabilidad.
  - Contingencia.

Los casos incluidos en el umbral definido por el administrador de aseguramiento de la calidad se ordenan por valor descendente del resultado de la evaluación de la función de similitud y se toma el primero como muestra de plan. A partir de ahí se analiza cada inspección para este caso y se busca en el resto de los casos seleccionados el que mejores resultados de las métricas tenga para esa inspección particular, con esta información se complementa la lista de chequeo de cada inspección para el proyecto que se está planificando.

El usuario del sistema en el momento en que define las políticas de la empresa que han sido consideradas en el desarrollo de su proyecto debe indicar cuáles de ellas deberán ser consideradas como imprescindibles. Las políticas imprescindibles son aquellas que tienen que estar presentes en el caso que se proponga, por tanto son las más importantes y deben tener mayor peso en la función de similitud.[15]

El caso propuesto será enriquecido por el administrador de aseguramiento de la calidad y la versión definitiva del plan de revisiones para el proyecto será almacenada en la base de datos del sistema y en la base de casos. Es importante destacar que la Base de Casos se actualiza también en el momento en que se documentan los resultados finales de una inspección, en este caso el sistema analiza la efectividad de la inspección teniendo en cuanta las métricas y decide si colocar la información correspondiente en un caso ya existente, modificando una inspección anterior o creando una nueva, o si por el contrario, solamente incorporará la información de la inspección en el registro detallado de seguimiento de las inspecciones de cada software en la base de datos.

CONCLUSIONES

La implantación de un sistema de calidad en una organización ayuda a mejorar la gestión del desarrollo de software, y esto traerá como consecuencia una disminución de los problemas y errores, favoreciendo las relaciones y comunicación entre las personas y grupos de organización, y de estos con los clientes o usuarios.

Planificar un Sistema de Evaluación de Calidad para cada proyecto contribuye considerablemente a elevar la calidad del producto final, considerando como aspecto muy importante el nivel de satisfacción del cliente. La planificación y realización de inspecciones durante el proceso de desarrollo de los proyectos es un aspecto a tener en cuenta para el logro de este objetivo, sobre todo si se realizan en los momentos adecuados.

Realizar estas inspecciones en las etapas tempranas puede contribuir no solo a detectar posibles fallas u omisiones en los requerimientos sino también a garantizar la completitud de la información para etapas posteriores del proceso de desarrollo. Contar con una propuesta de procedimiento, momento y aspectos en la evaluación del producto durante la inspección es el resultado de este trabajo y del software que en él se describe. El sistema permite controlar y dar seguimiento al plan de revisiones da cada proyecto, así como hacer un análisis de la calidad a partir de las métricas empleadas, haciendo uso de la técnica de razonamiento basado en casos, para llevar a cabo algunas de las responsabilidades del software.

REFERENCIAS

1. **PRESSMAN, R.:** "Software Engeneering, Software Engeneering Project Manager", *IEEE*, Computer Society, 1997.
2. **————.:** *Software Engeneering a Practitioner's Approach*, McGraw Hill, 4ta. ed., 1997.
3. **NORRIS, M.:** *The Healthy Software Project: A Guide to Successful Development And Management*, Wiley & Sons, 1993.
4. **SCHULMEYER, G:** *Handbook of Software Quality Assurance*, Prentice, May, 1997.
5. **KAN, S.:** *Metrics and Models in Software Quality Engineering*, Addison Wesley Longman, Inc., 1995.
6. **REYNOLDS, G:** *Information Systems for Managers*, West Publishing Company, 3ra. ed., 1995.
7. **FARLEY, R. :** *Ingeniería del Software*, McGraw Hill, 1990.
8. **LAGE, C.:** "Discurso, Informática 2000", Palacio de Convenciones, Ciudad de La Habana, Cuba.
9. *Lineamientos estratégicos para la informatización de la sociedad cubana*, Ciudad de La Habana, Cuba, 1997.
10. **FEBLES, A.:** "CASE Corporativo para el proceso de control de cambios", Tesis presentada en opción al título de Máster en Informática Aplicada, Instituto Superior Politécnico José Antonio Echeverría, Ciudad de La Habana, 2001.
11. **RUMBAUGH, J.:** *The Unified Modeling Language. Reference Manual*, Addison Wesley Longman Inc., 1999.
12. **KOLODNER, J.:** *Case-Based Reasoning*, Morgan Kaufmann Publishers Inc., 1993.
13. **SCHANK, R.:** *Inside Case-Based Explanation*, by Lawrence Erlbaum Associates Inc., 1994 .
14. **ALTHOFF, K.:** *Case-Based Reasoning. Handbook of Software Engineering and Knowledge Engineering*, Fraunhofer Institute for Experimental Software Engineering (IESE), Kaiserslautern, Alemania, 2001.
15. **MANJARES, A.:** "Razonamiento basado en casos", Universidad Nacional de Educación a Distancia, Departamento de Inteligencia Artificial, Madrid, España, 2001.
16. **WATSON, I.:** *Applying Case-Based Reasoning: Techniques for Enterprise Systems*, Morgan Kaufmann Publishers, Inc., 1997.
17. **ÁLVAREZ, S.:** "ADOOSIUML", Centro de Estudios de Ingeniería de Sistemas, Instituto Superior Politécnico José Antonio Echeverría, Ciudad de La Habana, Cuba, 2001.
18. **————.:** "METVISUALE", Centro de Estudios de Ingeniería de Sistemas, Instituto Superior Politécnico José Antonio Echeverría, Ciudad de La Habana, Cuba, 1999.
19. **FOWLER, M.:** *UML Destilled*, 2da. ed., Addison Wesley Longman Inc, 2000.
20. **JACOBSON, I.:** *El proceso unificado de desarrollo de software*, Addison Wesley Longman Inc., 2000.
21. **LARMAN, G:** *UML y Patrones*, Prentice Hall Hispanoamericana, SA, 1999.
22. **BOOCH, G:** *The Unified Modeling Language. User Guide*, Addison Wesley Longman Inc, 1999.