UNIVERSIDAD DE CASTILLA LA-MANCHA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA (CIUDAD REAL)

PLANIFICACIÓN Y GESTIÓN DE SISTEMAS DE INFORMACIÓN

ESTÁNDAR IEEE 1219 DE MANTENIMIENTO DEL SOFTWARE

Autor: Juan Angel Martínez Párraga
Profesor: Francisco Ruiz González
Fecha: Mayo de 1999

INDICE

INTRODUCCIÓN

"Este estándar describe un proceso iterativo para la gestión y ejecución de las actividades de Mantenimiento del Software. [...] Los criterios establecidos se aplican tanto a la planificación del Mantenimiento del Software mientras este está en desarrollo, como a la planificación y ejecución de las actividades de Mantenimiento para productos software existentes..."

De esta forma se describe el alcance del estándar IEEE 1219. Aunque sólo menciona las fases de desarrollo y de producción de un producto software, las fases que se describen en el interior de sus páginas cubren todo el ciclo de vida de un software, cualquiera que sea su tamaño o complejidad. Las fases del ciclo de vida mediante las que se dirige el estándar son:

- Identificación del Problema
- Análisis
- Diseño
- Implementación
- Pruebas del Sistema
- Pruebas de Aceptación
- Puesta en Producción o liberación de versión

Dentro de cada una de estas fases, el estándar define una serie de procedimientos que se han de llevar a cabo y con los que se identifican la documentación, personas y productos software que intervienen. Las etapas que se han de cubrir son:

- Entradas
- Procesos
- Controles
- Salidas

Como es lógico, dentro de todas las fases de desarrollo de un producto software, se llevan a cabo tareas que ya han sido definidas por el IEEE en estándares, al igual que la que nos ocupa, por lo que podemos encontrar múltiples referencias a estos estándares para su correcta ejecución. A continuación se detalla una tabla con estas referencias:

ESTÁNDAR	NOMBRE	FECHA
IEEE 612.12	Standard Glossary of Software Engineering Terminology	1990
IEEE 730	Standard for Software Quality Assurance Plans	1989
IEEE 828	Standard for Software Configuration Management Plans	1990
IEEE 829	Stasdard for Software Test Documentation	1983
IEEE 982.1	Standard Dictionary of Measures to Produce Reliable Software	1988
IEEE 982.2	Guide for the Use of IEEE Standard Dictionary of Measures to Produce Reliable Software	1988
IEEE 983	Guide for Software Quality Assurance Planning	1986
IEEE 1012	Standard for Software Verification and Validation Plans	1986
IEEE 1028	Standard for Software Reviews and Audits	1988
IEEE 1042	Guide to Software Configuration Management	1987
IEEE 1058	Standard for Software Prroject Managements Plans	1987
IEEE 1074	Standard for Developing Software Life Cycle Processes	1991

MANTENIMIENTO DEL SOFTWARE

Según el IEEE, el Mantenimiento del Software es la modificación de un producto software después de su entrega al cliente o usuario para corregir defectos, para mejorar el rendimiento u otras propiedades deseables, o para adaptarlo a un cambio de entorno. Sin embargo, y aunque no se aprecia a lo largo del estándar IEEE 1219, el proceso de Mantenimiento del Software comienza con las primeras fases del ciclo de vida, puesto que el coste de Mantenimiento va a estar tremendamente influido por las decisiones que se tomen en cada una de estas fases.

Existen diversos tipos de Mantenimiento del Software dependiendo de las demandas de los usuarios del producto Software a mantener:

- Mantenimiento Adaptativo
- Mantenimiento Correctivo
- Mantenimiento Perfectivo
- Mantenimiento Preventivo

MANTENIMIENTO ADAPTATIVO

Es la modificación de un producto software, después de su puesta en producción, para mantener operativo un programa mientras se realiza un cambio en el entorno de producción.

Tiene por objetivo la modificación de un programa debido a cambios en el entorno, bien cambios en el hardware o en el software, en el que se ejecuta.

En cuanto a los cambios en el hardware cabe destacar, por ejemplo, los cambios en la plataforma de instalación, pasar de un sistema de 16 bits a uno de 32 bits; los cambios en el sistema de comunicaciones, pasar de un sistema de trabajo stand-alone a un sistema de red; etc.

En cuanto a los cambios en el software cabe destacar, por ejemplo, cuando se desea pasar un sistema Cliente-Servidor típico a un sistema Three-Tier; los cambios de un cliente clásico desarrollado en Visual Basic, Delphi o C/C++ a un cliente "ligero" (*Thin Client*) para ejecutarse sobre un navegador de Internet; etc.

Este tipo de mantenimiento es el más usual debido a los rápidos cambios que se producen en la tecnología informática, que en la mayoría de ocasiones dejan obsoletos los productos software desarrollados, no por su inoperancia, sino por la competitividad entre las empresas, en las que cada vez influye más el software utilizado. Por ejemplo, dar acceso a los productos a través de Internet, etc.

MANTENIMIENTO CORRECTIVO

Es la modificación de un producto software después de su puesta en producción y para corregir los fallos descubiertos.

El Mantenimiento Correctivo tiene por objetivo localizar y eliminar los posibles defectos de los programas. A pesar de las pruebas y verificaciones que aparecen en etapas anteriores del ciclo de vida del software, los programas pueden tener defectos. Un defecto en un sistema es una característica del sistema que podría ser la causa de un fallo. El fallo se produce cuando el comportamiento del sistema es diferente al esperado por su especificación.

Estos fallos pueden ser de procesamiento, de rendimiento, de programación o de documentación. Según estudios realizados la mayoría de los defectos se originan en las fases de especificación de requisitos y de codificación, por lo que también son importantes las primeras fases del ciclo de vida para el Mantenimiento del Software.

MANTENIMIENTO DE EMERGENCIA

Es un mantenimiento correctivo realizado sin planificación previa y utilizado para mantener operativo el sistema.

MANTENIMIENTO PREVENTIVO

Este tipo de mantenimiento no se menciona como tal en el estándar IEEE 1219, sino que se incluye como Mantenimiento Perfectivo, pero que es conveniente tener en cuenta debido a los costes que puede ahorrar frente a otro tipo de mantenimientos.

Consiste en la modificación del producto Software sin alterar las especificaciones del mismo, para mejorar las propiedades de mantenimiento del producto y facilitar así las futuras tareas de mantenimiento.

Los cambios que se llevan a cabo son en cuanto a los comentarios del código, la reestructuración de los programas para mejorar su comprensión, etc.

MANTENIMIENTO PERFECTIVO

Es la modificación de un producto software, después de su puesta en producción y para mejorar el rendimiento o la mantenibilidad, que es la facilidad de mantenimiento que tiene un software, es decir, la facilidad de un software para ser modificado, lo que influye directamente en los costes del mantenimiento.

También suelen ser debidos a cambios en las especificaciones del producto, detalle que no ha tenido en cuenta el estándar IEEE 1219 al definir los tipos de mantenimiento, aunque si lo especifica a lo largo del texto.

En cuanto a los cambios para mejorar el rendimiento pueden ser, por ejemplo, la inclusión de un monitor transaccional y su gestión, para mejorar los accesos a la Base de Datos; la utilización de caches en los clientes de un sistema Cliente-Servidor, para liberar la carga de la red de comunicaciones; etc.

Si hablamos de cambios por modificación de los requisitos de usuario o de las especificaciones funcionales, podemos citar el aumento del número de usuarios del producto, lo que podría repercutir en el rendimiento del sistema,o el aumento de las funcionalidades, etc.

Las tres preguntas claves en el Mantenimiento del Software son:

1. ¿Existe alguna forma de determinar analíticamente los efectos de un cambio antes de su implementación?
2. ¿Existe alguna forma de garantizar que el cambio propuesto no tendrá efectos adversos en partes del sistema que no se han modificado?
3. ¿Se puede hacer una estimación razonable del esfuerzo total necesario para realizar una modificación antes de llevarla a cabo?

DEFINIENDO EL MANTENIMIENTO

La distinción entre las actividades asociadas con el desarrollo y aquellas asociadas con el mantenimiento, están basadas en la visibilidad de los problemas del sistema y la forma de transmitirselo a los usuarios, grupo de pruebas y desarrolladores. Antes del desarrollo, la visibilidad de los problemas del sistema para los usuarios y el grupo de pruebas podría ser relativamente baja, ya que la reparación de algunos problemas puede ser manejada de una manera informal por el grupo de desarrollo. Sin embargo, una vez que el sistema ha sido liberado, es necesario un proceso de seguimiento más formal. Esta formalidad es necesaria debido a que muchos grupos dentro de la organización influyen en la información transmitida de los problemas del sistema. Por ejemplo, el personal de soporte al usuario necesita saber sobre los problemas para que a su vez puedan describírselos a los usuarios. El grupo de pruebas necesita saber sobre los problemas ya que pueden ser requeridos rápidamente para generar un plan de pruebas enfocado sobre las áreas problemáticas. También los gerentes necesitan conocerlos para poder priorizar el trabajo que se está haciendo.

Una simple definición de mantenimiento es: el proceso de registro y seguimiento de problemas y la petición y gestión de respuestas. Un problema no necesariamente implica un defecto del software. Los problemas aparecen simplemente porque el comportamiento esperado del sistema no coincide con el comportamiento actual. Esto puede ser el resultado de un defecto del software, pero puede ser fácilmente el resultado de un desconocimiento de las capacidades del sistema, o de la utilización incorrecta del flujo de trabajo, o de una documentación pobre, o de cambios en las directrices de la empresa, o de otras tantas razones. La gestión de las respuestas a los problemas del sistema cubre un millar de posibles actividades. Algunas de las posibles respuestas a los problemas del sistema podrían incluir: liberación de nuevas versiones, sesiones de planificación, identificación de la formación necesaria, actualización de la documentación, etc.

Desde el punto de vista del grupo de desarrollo, el aspecto más significativo de un problema será si es causa de un defecto del software. La calidad de la información comunicada desde el usuario al grupo de desarrollo es crucial para poder determinarlo y para su rápida resolución.

ESTANDARES DE MANTENIMIENTO

Existen diversos estándar de otros tantos organismos internacionales de estandarización que tienen una relación directa o indirecta con el Mantenimiento del Software:

- Para los procesos del ciclo de vida del software: IEEE 1074 e ISO 12207.
- Para la calidad del software y sus métricas: IEEE 1061 e ISO 9126.
- Para el Mantenimiento del Software: IEEE 1219 e ISO/IEC 14764

Los estándares para los procesos del ciclo de vida del software nos permiten enccajar y asociar el proceso de mantenimiento con los demás procesos existentes para el software. Los estándares de calidad del software interesan en mantenimiento del software porque los factores de calidad del software (especialmente la complejidad y la mantenibilidad) inciden directamente sobre el esfuerzo de mantenimiento necesario.

IEEE 1074

El IEEE 1074-1995, *Developing Software Life Cycle Processes*, detalla el conjunto de actividades que aparecen obligatoriamente en el desarrollo y mantenimiento del software. La clasificación se realiza dependiendo de que los procesos sean de gestión de proyectos, antes del

desarrollo, durante el desarrollo, después del desarrollo o durante todo el ciclo de desarrollo de un producto software.

ISO 12207

En este estándar publicado en 1995, ISO/IEC 12207 *International Standard fro Information Technology – Software Life Cycle Processes*, se define el proceso de mantenimiento como una parte principal del ciclo de vida del software. En el se definen los procesos, actividades y tareas presentes en la adquisición, suministro, desarrollo, operación y mantenimiento del software.

IEEE 1061

El IEEE 1061 *Standard for a Software Quality Metrics Methodology* provee una metodología para establecer requerimientos de calidad e identificar, implementar, analizar y validar métricas de calidad de productos y procesos software. La metodología es aplicable a todo el software en todas las fases de cualquier estructura de ciclo de vida. En este estándar se establece la mantenibilidad como uno de los factores de la calidad del software.

ISO 9126

En la nueva revisión de 1998 del estándar ISO 9126 *Software Quality Characteristics and Metrics*, se abordan las características que determinan la calidad del software, tanto del producto como de los procesos para desarrollarlo y mantenerlo.

IEEE 1219

Este es el estándar objeto del documento, IEEE 1219 *Standard for Software Maintenance*. Hasta 1998, único estándar que integramente se ocupa del proceso de Mantenimiento del Software. En él se detalla un proceso iterativo para gestionar y realizar las actividades de mantenimiento.

ISO/IEC 14764

Este es el estándar específico sobre Mantenimiento del Software que publico ISO en 1998. Como todos los estándar, el acceso de su lectura está restringidos a aquellos que son miembros de la organización, o pagan por hacerlo.

OTROS ESTÁNDAR QUE TRATAN EL MANTENIMIENTO

Aunque la mayoría de los estándar que se muestran a continuación, son de organizaciones y asociaciones no relacionadas directamente con la Ingeniería del Software, si hacen referencias al Mantenimiento del Software desde los distintos puntos de vista de cada organización.

ESTÁNDAR	NOMBRE
AECL CE-1001-STD REV.1	Standard for Software Engineering of Safety Critical Software
AIAA ANSI/AIAA R-013	Recommended Practice for Software Reliability
BSI BS-7738	Specification for Information Systems Products Using SSADM – (Structured Systems Analysis and Design Method)
DEF 00-56 Part 2/2	Safety Management Requirements for Defence Systems Containing Programmable Electronics Part 2: General

	Applications Guidance
German Process-Model (V-Model)	Software Life-Cycle Process Model (V-Model)
ESA PSS-05-0 ISS.2	ESA Software Engineering Standards
IEC 60601-1-4	Medical Electrical Equipment—Part 1: General Requirements for Safety-4. Collateral Standard: Programmable Electrical Medical Systems
IEC 60880	Software for Computers in the Safety Systems of Nuclear Power Stations
NCC	STARTS Purchasers' Handbook – Procuring Software-Based Systems
NIST FIPS PUB. 106	Guideline on Software Maintenance
NIST NISTIR 4909	Software Quality Assurance: Documentation and Reviews
NRC NUREG/BR-0167	Software Quality Assurance Program and Guidelines
RTCA DO-178B/ED-12B	Software Considerations in Airborne Systems and Equipment Certification
SEI CMU/SEI-91-TR-24	Capability Maturity Model for Software
SEI CMU/SEI-91-TR-25	Key Practices of the Capability Maturity Model
SIS TR 321	Systems Development Reference Model

FASES DEL DESARROLLO EN EL MANTENIMIENTO DEL SOFTWARE

Este estándar define cambios en un producto software a través de un proceso de mantenimiento dividido en fases. Este proceso es iterativo y en cascada, con una gran semejanza al ciclo de vida del desarrollo clásico. Estas fases se detallan a lo largo del estándar, indicando en cada una los elementos de los que se dispone al empezar, por ejemplo la documentación, las tareas que se han de realiar y de que manera para seguir fielmente los estándar, y por último el resultado de la fase, como documentación generada, código, etc. Estas fases son:

- Identificación y Clasificación del Problema o de la Modificación.
- Análisis.
- Diseño
- Implementación.
- Pruebas del Sistema.
- Pruebas de Aceptación.
- Liberación del Producto.

Según el estándar, el proceso de mantenimiento se inicia con una **Solicitud de Modificación** *(MR, Modification Request)*, que es un término genérico que incluye los formularios asociados con los informes de problemas o fallos, documentos para el control de cambios en la configuración, etc.

Cada una de estas fases tienen asociadas métricas de calidad según los diversos estándar definidos para ello, y que permiten de manera eficaz realizar el seguimiento y gestión del proceso de Mantenimiento del Software.

Además de las métricas, en cada fase se realizará un seguimiento o control de los procedimientos que se llevan a cabo, con el fin de documentar y monitorizar las modificaciones que se lleven a cabo. Para ello se mantendrá un repositorio con la identificación y descripción de los cambios.

IDENTIFICACIÓN, CLASIFICACIÓN Y PRIORIZACIÓN DEL PROBLEMA

En esta fase se identifican, clasifican y asignan una prioridad inicial a las modificaciones del software. Cada Solicitud de Modificación será evaluada para determinar su clasificación y prioridad. Esta contiene diversos cambios a realizar sobre el productos software que posteriormente se agruparán en bloques de implementación. La clasificación será identificada según los tipos de mantenimiento: correctivo, adaptativo, perfectivo y de emergencia.

Para mantener en todo momento una correcta organización y control del Mantenimiento es aconsejable hacer una revisión periódica de los elementos de modificación recibidos, de esta manera se pueden observar los más críticos y solicitados, y ayudaría a prevenir cualquier bloqueo de trabajo o parada en alguna de las fases del Mantenimiento por la falta de orientación.

Una vez recibido la Solicitud de Modificación, comienza el Mantenimiento del Software. En esta fase los procedimientos a seguir son:

- Asignación de un Número de Identificación.
- Clasificación del tipo de mantenimiento.
- Análisis de la modificación para determinar si se acepta, se deniega o se evalúa.
- Realizar una estimación preliminar de la magnitud de la modificación.
- Priorizar la modificación.
- Asignar Solicitudes de Modificación a bloques de tareas planificadas para su implementación.

Serán identificados tanto la Solicitud de Modificación como los procesos que se determinen realizar. Estos identificadores se mantendrán en un repositorio para su control.

La frecuencia y duración de las reuniones de revisión de la clasificación deberá ser dependientes del proyecto. Como guía, estas reuniones deberían ser consideradas como revisiones de estado en vez de revisiones técnicas. Si después de unas pocas sesiones las revisiones duran más de una hora, debería aumentarse su frecuencia, por ejemplo, semanalmente. Si esto sigue siendo insuficiente es posible que se deba a tres razones principales:

- Si la discusión se centra en mejoras del sistema (mantenimiento perfectivo) será porque es necesario analizar la magnitud del desarrollo antes que llegar a un nivel de mantenimiento sostenido.
- Si el sistema es de nuevo desarrollo y requiere un importante esfuerzo de mantenimiento justo después de su puesta en producción, la estrategia a seguir será clasificar las Solicitudes de Modificación por tipo de mantenimiento e integrarlas en conjuntos para compartir las mismas áreas de diseño. De esta forma se priorizará por el conjunto, en vez de por Solicitud de Modificación individual, con lo que conseguiremos minimizar la repetición de las tareas de diseño, codificación, pruebas y liberación.
- Si el sistema es antiguo, es posible que se considere reemplazarlo o rediseñarlo.

De esta forma vemos la importancia de clasificar las modificaciones a realizar para que puedan ser aplicadas a un sistema particular de la mejor manera posible y sin que afecte al sistema existente o a otras posibles modificaciones.

La prioridad de las Solicitudes de Modificación deberá ser asignada por alguno de los integrantes del proyecto: El autor de la solicitud o un representante, un usuario reconocido, un

experto del dominio, ingenieros de software dentro del proyecto o el jefe de proyecto. Para ello hay que considerar los siguientes criterios:

- Recursos estimados inicialmente según la facilidad/dificultad de implementación y el tiempo aproximado para llevarla a cabo según los recursos disponibles.
- Impacto esperado a los usuarios actuales y futuros, indicando las ventajas y desventajas.
- Asignación de un bloque de modificaciones planificadas para minimizar el impacto a los usuarios.

Cuando finalice esta primera fase, se podrá disponer de una Solicitud de Modificación validada y las tareas que se van a llevar a cabo para realizar los cambios solicitados. Para evidenciar la conclusión de esta fase, tendremos que recopilar en el repositorio lo siguiente:

- Relación de los problemas o nuevos requerimientos.
- Evaluación de los problemas o nuevos requerimientos.
- Clasificación del tipo de mantenimiento solicitado.
- Prioridad inicial.
- Datos de verificación (para el mantenimiento correctivo)
- Estimación inicial de los recursos necesarios para modificar el sistema existente.

En la fase de análisis se realizarán los estudios de impacto y costes a bajo nivel, pero para realizar la clasificación inicial es deseable comprobar los costes generales estimados. Ya que esto es un proceso iterativo, cabe la posibilidad de que la gestión de la prioridad cambie durante las siguientes fases.

ANÁLISIS

En esta fase se estudia la viabilidad y el alcance de las modificaciones, que ya tenemos clasificadas y priorizadas, así como la generación de un plan preliminar de diseño, implementación, pruebas y liberación del software. La información que se va a utilizar en esta fase proviene del repositorio y de la Solicitud de Modificación validada en la fase anterior, además de la documentación del proyecto y del sistema existente.

Una Solicitud de Modificación podría generar varios requisitos de funcionalidad, rendimiento, ***usabilidad***, ***fiabilidad***, ***comprensibilidad*** y ***mantenibilidad***, que podrán ser descompuestos en varios requisitos de software, base de datos, interface, documentación y hardware. Por lo tanto es necesario que participen tanto los solicitantes, desarrolladores y usuarios para asegurar que los elementos de la solicitud no sean ambiguos. Los términos de *usabilidad*, *fiabilidad*, *comprensibilidad* y *mantenibilidad*, y otros, son propiedades de un producto software que afectan a su calidad. La definición de los mismos podría ser la siguiente:

- ***Usabilidad:*** Facilidad de utilización por parte de los usuarios.
- ***Fiabilidad:*** Probabilidad de que el software no falle.
- ***Mantenibilidad:*** Facilidad de modificación de un software.
- ***Comprensibilidad:*** Facilidad de comprender el software.

Si la documentación no está disponible o no es suficiente, y el código fuente es la única representación fiable del sistema software, la ingeniería inversa es el método más recomendado. Este es el procedimiento por el cual a partir del análisis de un sistema software existente, se identifican sus componentes y las interrelaciones que existen entre ellos, así como su representación en un nivel de abstracción más elevado. Para el estándar, el análisis es un

proceso iterativo que tiene al menos dos componentes: el análisis de viabilidad y el análisis detallado.

ANÁLISIS DE VIABILIDAD

El análisis de viabilidad va a desembocar en un **Informe de Viabilidad** (*FR, Feasibility Report*), que contiene los siguientes elementos:

- Impacto de las modificaciones. Identificar los efectos bilaterales potenciales.
- Soluciones alternativas, incluyendo prototipado.
- Análisis de los requisitos de conversión.
- Implicaciones de Seguridad.
- Implicaciones de Robustez.
- Factores humanos
- Costes a corto y largo plazo.
- Beneficios de realizar las modificaciones

ANÁLISIS DETALLADO

Para realizar el análisis detallado, el estándar ofrece unos procedimientos a seguir en cuanto a los elementos a modificar, las pruebas a realizar y el desarrollo y liberación de la versión final.

Al identificar los elementos susceptibles de modificación en el análisis, se examinan todos los elementos generados en las fases del Mantenimiento del Software que están afectados, por ejemplo el software, las especificaciones, las bases de datos, la documentación, etc. Cada uno de estos elementos serán identificados y generados si fuera necesario, especificando las partes del producto que van a ser modificadas, los interfaces afectados, los cambios esperados por el usuario, el grado relativo y clase de experiencia necesaria del personal para hacer los cambios, y el tiempo estimado para completar la modificación.

En cuando a la estrategia de pruebas, está basada en los elementos de modificación. Se tendrán que definir los requisitos para al menos tres niveles de pruebas: las pruebas de elementos individuales, las de integración y las de aceptación del usuario final. En estos también se ha de incluir, para cada uno de estos niveles, los requisitos de las pruebas de regresión, que se realizan para validar que el código que se ha modificado no introduce errores que no existían antes de la actividad de mantenimiento. Al ser modificaciones hechas sobre un producto software ya desarrollado hay que revalidar las pruebas básicas de éste, por lo que se tendrán que generar y utilizar nuevos casos de prueba para el mismo.

Por último, se tiene que desarrollar un plan de implementación preliminar que tendrá que prever el menor impacto posible para los usuarios del sistema software, tanto desde el punto de vista del futuro trabajo con el sistema, como durante la fase de desarrollo, para que esta no sea con métodos intrusivos en el entorno en producción.

En resumen, podemos decir, que el análisis detallado contempla los siguientes pasos:

- Definición de los requisitos fijados para la modificación
- Identificación de los elementos a modificar.
- Identificación de los elementos de seguridad.
- Identificación de los elementos de robustez.
- Definición de la estrategia de pruebas.
- Desarrollo de un plan de implementación.

La gestión en la fase de análisis tiene que realizarse metódicamente, obteniendo, en primer lugar, la última versión de la documentación del proyecto y del sistema ha modificarse que debe tener la propia organización, y verificando que toda la documentación del análisis y del proyecto sea actualizada y controlada. En cuanto a las funciones de pruebas asignadas a la organización, hay que verificar que exista la estrategia apropiada para la realización de estas pruebas y que la planificación de los cambios pueda soportar la realización de las mismas.

Los analistas tendrán que revisar los cambios propuestos para poder documentar la contribución técnica necesaria y la viabilidad económica de las modificaciones. Una parte importante será la identificación de los elementos que afectan al control de la seguridad y la robustez del sistema, para comprobar como afectan los cambios en ellos. También, como parte de este control, se revisarán los recursos estimados y la planificación temporal, para poder verificar su perfecto cumplimiento.

En el Mantenimiento del Software, el principal efecto se produce por realizar modificaciones en un sistema software en producción, por lo que habrá que considerar la integración de los cambios propuestos dentro del sistema existente, como uno de los principales factores de riesgo.

El objetivo final del análisis será la generación de la documentación necesaria, por lo que se debe realizar una revisión técnica y realizar los informes con los problemas existentes y las mejoras propuestas que van a ser implementadas en la nueva versión del producto software.

Como resumen podemos relacionar los documentos que deben incluirse en el repositorio tras la fase de análisis:

- Informes de viabilidad para las Solicitudes de Modificación recibidas.
- Informe de análisis detallado
- Actualización de los requisitos
- Lista de modificaciones preliminares
- Estrategia de pruebas
- Plan de implementación.

DISEÑO

A partir de toda la documentación existente del proyecto y del sistema que esté en producción, así como todo el código fuente y las bases de datos de la última versión, se va a realizar el diseño de las modificaciones del sistema en base a la documentación generada en la fase de análisis (análisis detallado, informe de requisitos, identificación de los elementos afectados, estrategia de pruebas y plan de implementación).

El primer paso en la fase de diseño será identificar los módulos software que van a ser objeto de modificación, con el fin de hacer constar la planificación de tareas y ver la previsión de las mejoras a introducir. Según se avanza en los módulos se realizarán las modificaciones oportunas a la documentación de los mismos. Esta documentación consiste de diagramas de flujo y control, esquemas, etc.

Para las modificaciones a realizar, se generará unos casos de pruebas que incluyan los elementos de seguridad y robustez. Además hay que identificar e incluir las pruebas de regresión necesarias.

En la documentación que se va generando en la fase de diseño, se tendrán que identificar y documentar los cambios que se realicen sobre los requisitos, y mantener al día una lista con las modificaciones que se van a llevar a cabo.

Como forma de asegurar el cumplimiento de los estándar, se ha de realizar la inspección del diseño de acuerdo con el estándar IEEE 1028-1988, y verificar que la documentación de los requisitos y del nuevo diseño está de acuerdo a una **Autorización de Cambio del Software** (*SCA, Software Change Authorization*), según el estándar IEEE 1042-1987.

Además, se ha de verificar la inclusión en el repositorio del nuevo material de diseño, sin olvidar los elementos de seguridad y robustez, cuidando de que la documentación de pruebas haya sido actualizada.

Como último paso en la verificación del cumplimiento de los estándares de calidad, se ha de comprobar el cumplimiento y coherencia de los requisitos en la fase de diseño.

La fase de diseño no se dará por finalizada hasta no disponer o asegurar los siguientes elementos:

- Lista de modificaciones revisada.
- Guía básica del diseño actualizada.
- Planes de pruebas actualizados.
- Análisis detallado actualizado.
- Requisitos verificados.
- Plan de implementación revisado.
- Lista de restricciones y riesgos bien documentados.

La fase de implementación debería comenzar durante la fase de diseño para comprobar la factibilidad de los cambios propuestos. Es posible que el grupo de ingeniería no entienda completamente el impacto y magnitud de los cambios hasta que no finalice la fase de diseño, o puede que un cambio específico sea demasiado complejo de implementar.

El proceso de diseño puede variar de un proyecto a otro y dependerá de: las herramientas utilizadas, el tamaño de las modificaciones, el tamaño del sistema existente, la disponibilidad o no de un sistema desarrollado y la accesibilidad a los usuarios y la organización que ha solicitado el mantenimiento.

Para asegurar la fiabilidad y mantenibilidad del sistema software, hay que tener presente desde las primeras fases del ciclo de desarrollo todas las características del producto, ya que las decisiones de diseño sobre las modificaciones de módulos software van a afectar a la calidad del propio software.

IMPLEMENTACIÓN

Para dirigir apropiadamente el esfuerzo en la fase de implementación será necesario disponer, como en las otras fases, de la documentación actualizada del proyecto y del sistema, del código fuente actual, y los resultados de la fase de diseño. Otros elementos necesarios para el éxito de esta fase serán:

- Documentación de diseño y requisitos aprobados y controlados.
- Estándar de codificación acordado para ser utilizado por el grupo de mantenimiento.
- Métricas de diseño que podrían ser aplicables en la fase de implementación (podrían clarificar la complejidad del código a desarrollar o mantener).

- Una planificación de desarrollo detallada, incluyendo todas las revisiones de código que se harán y a que nivel.
- Un conjunto de respuestas a los riesgos identificados en las fases anteriores y que serán aplicables en la fase de pruebas.

En esta fase, se van a seguir unos determinados procesos que serán iterativos, y gradualmente incrementales, es decir, se irán repitiendo y desarrollando en mayor detalle, hasta obtener el resultado previsto por la organización.

Estos procesos son:

- Codificación y pruebas de unidad.
- Integración.
- Análisis de riesgo y revisión.
- Revisión de disponibilidad de pruebas.

CODIFICACIÓN Y PRUEBAS DE UNIDAD

Mediante este proceso se implementan los cambios en el código y se realizan las pruebas de unidad, así como otros procesos de verificación y validación y aseguramiento de la calidad.

INTEGRACIÓN

Tras la codificación y las pruebas de unidad, o incluso durante la fase de codificación, el software modificado puede ser integrado con el sistema existente, pudiendo a su vez, realizar el refinamiento de las pruebas de integración y regresión. También, como objetivo, se pretenden valorar todos los efectos producidos tras la modificación del software existente, ya que cualquier impacto inaceptable debe ser conocido, para poder volver a la fase de codificación y pruebas y corregir su efecto.

REVISIÓN Y ANÁLISIS DE RIEGOS

Durante la fase de implementación, la revisión y análisis del riesgo será realizada periódicamente, siempre durante la fase, preferiblemente a su final, al igual que en las fases de análisis y diseño. Es lo más recomendado ya que un alto porcentaje de los riesgos y problemas de diseño, costes y rendimiento aparecen mientras se realiza la modificación del sistema.

Para cuantificar el análisis del riesgo, se utilizarán las métricas oportunas según los estándar. Llega a ser especialmente importante sí el número de iteraciones de la codificación y las pruebas de unidad, así como la integración, están fuera del límite establecido al iniciar las modificaciones. Si es así, será necesario re-evaluar la factibilidad del diseño y/o modificación de los cambios solicitados, y volver a la fase de diseño, análisis o incluso de identificación y clasificación.

REVISIÓN DE LA DISPONIBILIDAD DE PRUEBAS

Para valorar la disponibilidad de las pruebas del sistema, se mantendrá una revisión de acuerdo con el estándar IEEE 1028-1988 de revisión y auditoría del software. Antes de comenzar la revisión, se tendrá que preparar y facilitar al personal del proyecto la siguiente información:

- Criterios iniciales para las pruebas del sistema
- Recursos y tiempo necesario para su realización.
- Plan de pruebas detallado.
- Plantillas de documentación de las pruebas.
- Procedimientos para resolver anomalias.
- Procedimientos para la Gestión del Configuración del Software.
- Criterio para los resultados de las pruebas de sistema.
- Resultados de las pruebas a bajo nivel.

Cuando se esté en la fase de implementación de los cambios pedidos en la Solicitud de Modificación, se debe asegurar, al igual que en el resto de fases, de que se cumple con los estándar IEEE desarrollados a tal efecto. Así, por ejemplo, para cumplir el estándar IEEE 1028-1988 de revisión y auditoría del software, se ha de inspeccionar periódicamente el software que se va desarrollando con el fin de comprobar que el código se ajusta al estándar de codificación definido en la fase de implementación, y de esta menera aumentar la comprensibilidad del código.

Todas las pruebas de integración y de unidad deben estar documentadas, por lo que será necesario crear un archivo o carpeta en el repositorio, especifico para el desarrollo y la documentación de las pruebas. Habrá que asegurarse de que efectivamente estas pruebas y su documentación se realice según los estándares, además de comprobar que la documentación técnica y de usuario también han sido actualizadas.

Durante las sucesivas revisiones del software y de las pruebas, como hemos mencionado anteriormente, pueden aparecer riesgos debido a los cambios que se van a producir en el software existente. Estos riesgos deben ser identificados y documentados e incluso resueltos cuando proceda.

Por último, el resultado de la fase de implementación, será un nuevo software que deberá estar bajo el control de la **Gestión de Configuración del Software** (*SCM, Software Configuration Management*), al que se acompañará de toda la documentación actualizada de diseño, documentación de pruebas, documentación de usuario y material de aprendizaje, sin olvidarnos de la declaración de riesgos e impacto para los usuarios y el informe de revisión de la disponibilidad de pruebas.

PRUEBAS DEL SISTEMA

Las pruebas de sistema se realizan sobre un sistema modificado. Deberían ser realizadas por una organización independiente y siempre estar presente el cliente y el usuario final. La organización responsable de las pruebas del sistema deberá ser independiente de los desarrolladores y diseñadores del software, pero podrían utilizarse como recursos para el personal de pruebas.

Una parte de estas pruebas son las de regresión, que se realizan para validar que el código que se ha modificado no introduce errores que no existían antes de la actividad de mantenimiento. Si se han realizado cambios al software o a los casos de prueba una vez que el producto ha sido liberado, será necesario realizar las pruebas de unidad y regresión durante la fase de análisis para establecer la línea de referencia del producto, que son unos datos básicos utilizados como referencia en diversas métricas, de rendimiento, calidad, etc.

Para realizar las pruebas de sistema dispondremos de toda la documentación generada a tal efecto en las fases anteriores que son:

- Informe de revisión de disponibilidad de pruebas
- Plan de pruebas de sistema
- Casos de pruebas de sistema
- Procedimientos de prueba del sistema
- Manuales de usuario
- Diseño

Esta fase ha de llevarse como si se tratara de un sistema completamente integrado. Los procesos que han de realizarse durante las pruebas de sistema son:

- Pruebas de funcionalidad del sistema
- Pruebas de interface
- Pruebas de regresión
- Revisión de la disponibilidad de pruebas del sistema para valorar el nivel de preparación para las pruebas de aceptación.

El resultado de las pruebas que se hayan realizado antes de la revisión de la disponibilidad de pruebas, no deberían ser utilizadas como parte del informe de las pruebas del sistema para comprobar los requisitos a nivel de sistema. Esto es necesario para asegurar que la organización no considera que el probar una por una todas las partes del sistema constituye una "Prueba del Sistema", tal como se concibe esta fase. El cliente participará en la revisión para ver que la versión de mantenimiento está lista para empezar las pruebas de aceptación.

Las pruebas del sistema deberán ser conducidas como una función de pruebas independiente, o por la función de aseguramiento de calidad del software. Antes de finalizar esta fase, la función de pruebas será responsable de informar del estado del criterio que se ha establecido en el plan de pruebas para satisfacer la correcta finalización de las pruebas del sistema, es decir, especificar los criterios seguidos para decir que el sistema ha superado o no con éxito las pruebas.

Todo el código fuente, las Solicitudes de Modificación y la documentación de pruebas, será puesta bajo la Gestión de Configuración del Software, de acuerdo con el estándar IEEE 828-1990. Esta misma función será la que realizará el control de las versiones del software y todos los ficheros durante las pruebas del sistema. Los controles necesarios para asegurar la integridad del producto serán realizados por la función de Aseguramiento de la Calidad del Software, de acuerdo con el estándar IEEE 730-1989, que procurarán que los cambios a los productos que son liberados, están autorizados y son técnicamente correctos.

Al acabar la fase de pruebas del sistema tendremos un sistema completamente integrado y probado, junto a los informes de pruebas y el informe de revisión de la disponibilidad de pruebas.

Para el mantenimiento de futuras versiones, es posible que se realicen otras pruebas para satisfacer requisitos en los que se necesite interactuar con otros sistemas o subsistemas. En este caso será necesario controlar y validar que no se introduzcan nuevos fallos como resultado de los cambios.

PRUEBAS DE ACEPTACIÓN

Al igual que las pruebas de sistema, las pruebas de aceptación serán realizadas sobre un sistema completamente integrado. Estas pruebas se realizan por el cliente, por el usuario o por

un tercero designado por el cliente. Se llevan a cabo para asegurar que el resultado de las modificaciones es satisfactorio para el cliente, tanto del software como de la documentación generada.

Las pruebas de aceptación siempre han de realizarse con software que esté bajo la Gestión de Configuración del Software, de acuerdo con los estándar IEEE 828-1990 y IEEE 730-1989.

Para comenzar las pruebas de aceptación, hemos de asegurarnos de disponer del informe de revisión de disponibilidad de pruebas, así como del sistema totalmente integrado. En las fases anteriores han de haberse preparado los planes para las pruebas de aceptación y los casos y procedimientos para realizar estas pruebas.

Los resultados de las pruebas realizadas antes del informe de revisión de disponibilidad de pruebas prodrían ser utilizados por el cliente para reducir el alcance de las pruebas de aceptación. Si es así, el cliente deberá documentar, en el informe de las pruebas de aceptación, los resultados que se tomaron de anteriores pruebas.

Lo primero que se ha de realizar son las pruebas de aceptación a nivel funcional, para proseguir con las pruebas de interoperabilidad y las de regresión.

Antes de terminar las pruebas de aceptación, la organización encargada de realizarlas deberá responsabilizarse de informar pertinentemente sobre el estado de los criterios que se han establecido en el plan de pruebas para llegar a comprobar que se han superado con éxito las pruebas. El cliente o un representante del mismo formarán parte del grupo de revisión al que se le informará de estos criterios y evaluar así el resultado de las pruebas para asegurarse de que, la versión de mantenimiento, está lista para ser puesta en producción.

El papel que desempeña en esta fase aquel que realiza la Modificación del Software, será la de comprobar que se realizan las pruebas, realizar el informe de resultados de las pruebas para la **Auditoría de Configuración Funcional** (*FCA, Functional Configuration Audit*) basándose en el estándar IEEE 1028-1988, así como conducir la auditoría funcional.

Tras las pruebas se establecerá el nuevo sistema de referencia y se situará la documentación generada durante las pruebas de aceptación bajo el control de Gestión de Configuración del Software. Una parte de esta documentación será un informe final de las pruebas de aceptación, que deberá estar bajo el estándar IEEE 1042-1987 y cuyo responsable será el propio cliente.

De igual forma que con las pruebas del sistema, para las siguientes versiones de mantenimiento es posible que se realicen otras pruebas de aceptación con el fin de satisfacer los requisitos de interface con otros sistemas o subsistemas y con el fin de validar que no se han introducido fallos como resultado de los cambios.

PUESTA EN PRODUCCIÓN

Una vez probado completamente el sistema, estamos a punto de pasar a la fase de liberación de la versión modificada.

Los pasos a seguir para instalar la nueva versión serán los siguientes:

- Conducir una **Auditoría de Configuración Física** (*PCA, Physical Configuration Audit)*. Será necesario organizar y documentar la auditoría según el estándar IEEE 1028-1988.
- Notificar a la comunidad de usuarios.
- Desarrollar una versión de archivo del sistema para salvaguarda del mismo.
- Realizar la instalación y formación para el cliente. Se ha de proveer del material de sistema necesario para facilitar la utilización a los usuarios.
- Se ha de completar la **Documentación de Descripción de la Versión** (*VDD, Version Description Document)*, según el estándar IEEE 1042-1987.
- Hay que actualizar el estado de la base de datos de cuentas y auditoría.
- Y finalmente, hay que poner todo bajo el control de la Gestión de Configuración del Software.

Para reducir los riesgos asociados con la instalación de la nueva versión del sistema software, el jefe de proyecto debería planificarlo y documentar los procedimientos de instalación alternativos, que podrían asegurar el mínimo impacto sobre los usuarios del sistema debido a fallos del software imprevistos, no detectados durante las pruebas. El plan deberá incluir factores críticos en tiempo, por ejemplo, las fechas disponibles para la instalación, así como los procedimientos de recuperación o vuelta atrás.

Si las modificaciones afectan a cambios significativos en la documentación, se tendrá que estudiar la necesidad de la formación del usuario. Cuando las modificaciones del sistema afectan al interface de usuario o es una modificación de funcionalidad significativa, será necesario realizar una fase de formación del usuario, que pueden incluir métodos formales (demostraciones prácticas) y no formales.

Al estar bajo la Gestión de Configuración del Software se realizará una copia de seguridad de la versión del sistema existen y de la nueva versión. Para facilitar la recuperación se deberían archivar las copias del sistema completo en una localización distinta de la que este en producción. Esta copia de seguridad contendrá el código fuente, la documentación de requisitos, la documentación de diseño, la documentación de pruebas incluyendo los datos de los casos de prueba, y el entorno de soporte (sistema operativo, compilador,...).

PLANIFICACIÓN DEL MANTENIMIENTO

Al comenzar un proceso de mantenimiento, e incluso antes de comenzar un proyecto Software, es necesario realizar una planificación de las futuras etapas de mantenimiento, para determinar el esfuerzo humano, material y económico necesario para llevar a cabo un mantenimiento eficaz. Esta planificación deberá seguir el estándar de gestión de proyectos IEEE 1058-1987. El plan de mantenimiento puede incluir lo siguiente:

- Determinar el esfuerzo de mantenimiento
- Determinar el proceso de mantenimiento a seguir
- Cuantificar el esfuerzo de mantenimiento
- Determinar los requisitos del proyecto de mantenimiento
- Desarrollar un plan de mantenimiento.

DETERMINAR EL ESFUERZO DE MANTENIMIENTO

El primer paso en el proceso de planificación del mantenimiento es un análisis de los niveles de servicio y capacidades existentes. Esto incluye un análisis del plan de mantenimiento

actual, si existe, y el estado de cada uno de los sistemas dentro de este plan. A nivel de sistema, cada uno deberá ser examinado para determinar lo siguiente:

- Tiempo transcurrido desde la puesta en producción.
- Número y tipo de cambios durante su ciclo de vida.
- Utilidad del sistema
- Número y tipo de Solicitudes de Modificación recibidas
- Calidad y temporalidad de la documentación
- Estadísticas existentes del rendimiento (CPU, E/S disco, etc.)

Las descripciones a nivel del plan consistirán en la descripción del esfuerzo global y las necesidades del área de mantenimiento. Esto incluye las cantidades y clase de deficiencias en la funcionalidad del sistema dentro de la arquitectura del plan de mantenimiento existente.

Además, las revisiones del grupo de mantenimiento y de los procedimientos de mantenimiento son necesarias para determinar el esfuerzo de mantenimiento global El análisis a este nivel es simplemente para reunir aquellas medidas necesarias para determinar lo siguiente:

- Número de personas en el grupo de mantenimiento, descripción de sus tareas, y sus tareas actuales.
- Nivel de experiencia del grupo de mantenimiento, tanto para la aplicación actual como en la industria en general.
- La relación de gastos y las posibles razones para abandonar el mantenimiento.
- Los métodos documentados actualmente para mantenimiento a nivel de sistema y de programa.
- Los métodos utilizados actualmente por el grupo de programación.
- Las herramientas utilizadas para el soporte del mantenimiento y como se utilizan.

La información a este nivel es utilizada para definir la línea de referencia para la organización del mantenimiento y para proveer un entorno de valoración de las necesidades de cambio.

DETERMINAR EL PROCESO DE MANTENIMIENTO A SEGUIR

El proceso de mantenimiento es una forma natural de comprobar la obsolescencia de muchas de las medidas del sistema que se tienen como referencia. Una vez estas medidas han sido recogidas, es necesario determinar el siguiente paso en el mantenimiento. En algunas organizaciones el proceso es hecho a medida según el tipo de mantenimiento que se está realizando y puede ser dividido de diferentes maneras, que pueden incluir procesos diferentes para las correcciones o para las mejoras, para los pequeños cambios o para los grandes cambios, etc. Es útil clasificar las formas de iniciar el mantenimiento antes de definir los procesos en sí.

Cada uno de los procesos será descrito por una serie de eventos. En general, el flujo de trabajo se describe desde la recepción de la Solicitud de Modificación hasta su implementación y liberación.

CUANTIFICAR EL ESFUERZO DE MANTENIMIENTO

Cada uno de los pasos en el proceso necesita ser descrito numéricamente en términos de volumen de tiempo. Estos números pueden entonces ser utilizados como base para determinar el rendimiento actual de la organización de mantenimiento.

DETERMINAR LOS REQUISITOS DEL PROYECTO DE MANTENIMIENTO

A este nivel, el proceso de mantenimiento necesita ser asociado a un entorno de negocio. Se debería completar una revisión de las futuras expectativas, que podrían incluir:

- Cambios del sistema esperados, externos o regulatorios.
- Cambios internos esperados para soportar los nuevos requisitos.
- Lista de nuevas funciones y características deseadas.
- Mejoras esperadas en el rendimiento, adaptabilidad, conectividad, etc.
- Nuevas líneas de negocio que necesitan ser soportadas.
- Nuevas tecnologías que necesitan ser incorporadas.

Todo esto necesita ser cuantificado o dimensionado para determinar la carga de mantenimiento futuro para la organización.

DESARROLLAR UN PLAN DE MANTENIMIENTO

La información recogida dará una base para el nuevo plan de mantenimiento. El plan deberá cubrir cuatro áreas principales:

- Proceso de Mantenimiento
- Organización
- Reserva de Recursos
- Auditoría del Rendimiento.

Cada uno de estos elementos serán incluidos en el plan de mantenimiento final. El primer proceso deberá ser descrito en términos de su alcance, la secuencia de procesamiento, y la manera de controlar este proceso.

ALCANCE DEL PROCESO

El plan necesita definir las fronteras del proceso de mantenimiento. El proceso comienza en algún punto (recepción de la Solicitud de Modificación) y termina con alguna acción (puesta en producción). La diferencia entre el mantenimiento y el desarrollo se verá claramente en este punto: ¿Una mejora se considerada un nuevo desarrollo o un mantenimiento? ¿En que punto un sistema de nuevo desarrollo entre en el proceso de mantenimiento?

Otro elemento que debería ser definido dentro del alcance es si debe categorizarse el proceso de mantenimiento y cómo. ¿Habrá diferencias entre notificaciones y otros tipos de mantenimiento? ¿Se considerarán las adaptaciones y mejoras dentro del mismo proceso o se gestionarán de forma diferente?

SECUENCIA DEL PROCESO

La secuencia debería utilizar el proceso descrito en este estándar como una guía. Es necesario describir el flujo global de trabajo incluyendo lo siguiente:

- Entrada en la Gestión de Configuración del Software automatizada y los sistemas de gestión de proyectos.
- Descripciones de cada uno de los pasos del proceso y sus interfaces.

- Flujo de datos entre los procesos.

CONTROL DEL PROCESO

Cada uno de los pasos en el proceso debería ser controlado y medido. Deberían ser definidos los niveles de rendimiento esperados. Los mecanismos de control deberían ser automatizados en lo posible. El control de los procesos debería seguir el estándar que nos ocupa.

ORGANIZACIÓN

El tamaño del grupo de trabajo puede ser estimado según la carga actual de trabajo y las necesidades futuras estimadas. Esta estimación podría además estar basada en la productividad esperada de cada uno de los pasos del proceso.

RESERVA DE RECURSOS

Una parte importante del plan de mantenimiento es un análisis del hardware y el software más apropiado para soportar las necesidades de la organización. El desarrollo, mantenimiento, y las plataformas de destino deberán ser definidas. Se describirán las diferencias entre los entornos de trabajo. Se identificarán y proveerán las herramientas que se vayan a utilizar para mejorar la productividad. Estas herramientas tendrán que estar disponibles para todo aquel que las necesite, además de proporcionar el entrenamiento apropiado para su correcta utilización.

AUDITORÍA DEL RENDIMIENTO

Una vez iniciado el proceso de mantenimiento, se debería monitorizar y evaluar para juzgar su efectividad. Si cada uno de los pasos en el proceso tiene sus criterios de medida, se debería ser sencillo obtener y evaluar el rendimiento.

DESARROLLO DEL PLAN

El desarrollo de un plan de mantenimiento es llevado a cabo de la misma manera que cualquier cambio estructural. Es importante disponer de los recursos técnicos, profesionales y directivos que sean posibles.

GESTIÓN DE CONFIGURACIÓN DEL SOFTWARE

La Gestión de Configuración del Software (*SMC*) es un elemento crítico del proceso de Mantenimiento del Software y que ha de añadirse a este proceso, según los estándares IEEE 828-1990 y IEEE 1042-1987. Es un proceso dirigido por procedimientos para implementar la función de ingeniería del software. Estos procedimientos ofrecen la verificación, validación y certificación de cada uno de los pasos necesarios para identificar, autorizar, desarrollar y liberar el producto software. Además, definirán los métodos utilizados para controlar los cambios durante todo el proceso de mantenimiento, para ofrecer la facilidad de seguimiento del proyecto requerida, para asegurar la integridad del producto, para mantener periódicamente informado al jefe de proyecto, y para documentar todas las actividades que se realicen.

La gestión de la documentación durante las pruebas del sistema deberá ser realizada por la función de Gestión de Configuración del Software. Los documentos que se manejarán serán:

- Documentación del Sistema
- Código y listados del Software
- Solicitudes de Modificación
- Documentación de Pruebas

Aunque la Gestión de Configuración del Software no se involucra muy profundamente en las fases iniciales del desarrollo de software tradicional, éste se realizará activamente en todas las fases del proceso de Mantenimiento del Software. Las equivocaciones para dar una rigurosa Gestión de Configuración del Software pueden provocar el caos durante el Mantenimiento, por lo que identificaremos los elementos que se han de imponer en cada una de las fases del Mantenimiento del Software conforme al estándar IEEE 1219.

IDENTIFICACIÓN, CLASIFICACIÓN Y PRIORIZACIÓN DEL PROBLEMA

El proceso de Gestión de Configuración del Software es el principal elemento de la fase de identificación del problema en el Mantenimiento del Software. Este procedimiento es responsable de recibir y dar entrada al problema dentro del sistema de **Cuentas de Estado de Configuración** (*CSA, Configuration Status Accounting*).

El personal de la Gestión de Configuración del Software será responsable de dirigir el problema a la persona adecuada, por ejemplo al ingeniero de software, de sistemas o de pruebas, para su validación y evaluación. La Gestión de Configuración del Software ofrecerá el seguimiento y coordinación de la documentación del problema durante esta fase.

ANÁLISIS

Durante la fase de análisis, será responsabilidad de la Gestión de Configuración del Software ofrecer la documentación actualizada al personal que realiza el análisis. Además tendrá que facilitar los listados actualizados del código fuente y los informes del sistema de Cuentas del Estado de Configuración mostrando el estado exacto del problema.

Al acabar la fase de análisis, será responsabilidad de la Gestión de Configuración del Software de asegurar que los resultados del análisis sean presentados para su revisión, una tarea de la Gestión de Configuración del Software. Esta revisión dará una visibilidad y seguimiento dentro de la resolución del problema ya que se tendrá que decidir si continuar con el análisis o comenzar la fase de implementación. Los problemas que sean aprobados para su implementación, se asignarán a un paquete de versión de ingeniería del software. Aunque cada problema sea implementado independientemente, el control se realizará a un bloque de problemas asociados por la versión de ingeniería del software. Será responsabilidad de la Gestión de Configuración del Software documentar los procedimientos a seguir durante las revisiones y la actualización de los registros de la Cuenta de Estado de Configuración asociados al problema.

DISEÑO

Durante la fase de diseño, la Gestión de Configuración del Software se responsabilizará de asegurar que el personal encargado del diseño tenga la documentación actualizada de la librería del sistema. Será muy importante que los diseñadores reciban cualquier cambio en la documentación tan pronto como sea posible.

Otra de las responsabilidades será la de archivar y salvaguardar los datos provistos por el personal de diseño, tales como resultados de la inspección o revisión del software. En un entorno de diseño automatizado, por ejemplo si se utilizan herramientas CASE (*Computer-Aided Software Engineering*), es posible que la Gestión de Configuración del Software ofrezca asistencia para el mantenimiento y control de versiones. De hecho, siempre ofrecerá asistencia y guía al personal de diseño para la selección y utilización de las convenciones de nomenclatura.

Al final de la fase de diseño, tendrá que asegurarse que la documentación de diseño se encuentre en lugar seguro y esté bajo el control de la Gestión de Configuración del Software para proteger la integridad del resultado del diseño. Si se trata de sistemas automatizados, será responsable de asegurar el control riguroso de las versiones.

IMPLEMENTACIÓN

Durante la fase de implementación, la Gestión de Configuración del Software será responsable de ofrecer a los programadores copias de los módulos que se van a modificar y asegurarse del control riguroso de versiones. En las situaciones donde se realicen múltiples cambios a un solo módulo, también se responsabilizará a la Gestión de Configuración del Software de asegurar la organización convenientemente para prevenir la pérdida de los cambios. Esto requiere una completa cooperación y coordinación entre la Gestión de Configuración del Software y el equipo de mantenimiento. Se deberá notificar a este equipo cuando se dispongan de los requisitos o datos de diseño actualizados.

La Gestión de Configuración del Software será la responsable de mantener el control de la configuración sobre todas las herramientas de soporte utilizadas en el proceso de Mantenimiento del Software. El control de las herramientas, por ejemplo compiladores, sistemas operativos, ensambladores, etc., es crucial para evitar un mayor número de fuentes de error en el plan de mantenimiento y prevenir trabajo innecesario.

Al final de la fase de implementación se responsabilizará de recoger todos los módulos que hayan sido modificados, agruparlos en librerías y guardarlos en lugar seguro. Si se utiliza el concepto de *carpeta* en el desarrollo de software, la Gestión de Configuración del Software deberá asegurar que cada una de ellas se pone bajo el control de la configuración. A menudo va a ser responsable de regenerar el sistema y dejarlo disponible para el personal de pruebas. Esto será una buena práctica que asegurará unos resultados consistentes para el grupo de pruebas. Esta tarea la pueden realizar el equipo de jefes de programadores.

La Gestión de Configuración del Software será responsable de asegurar que todo los cambios planificados sean incluidos en la versión paquetizada, y que estén disponibles para las pruebas del sistema. La versión completa estará sujeta a la Auditoría de Configuración Física y validada por el Aseguramiento de la Calidad del Software. La auditoría verifica y valida que todos los elementos están completos (documentación actualizada, procedimientos de prueba, descripciones de versión, etc.). Después de acabar la auditoría con éxito, la versión paquetizada será presentada a revisión para su aprobación y proceder con las pruebas de sistema. Este proceso de aprobación ayudará a reducir la pérdida de recursos del sistema para pruebas cuando el producto no esté listo, dando un substancial beneficio de coste y tiempo.

Por último se responsabilizará de la actualización de la base de datos de la Cuenta de Estado de Configuración con los resultados tomados en la revisión de la versión, y la generación de informes del estado para el equipo de mantenimiento y gestión. Los resultados e informes generados por la auditoría serán archivados y formarán parte de la documentación de la versión.

PRUEBAS DEL SISTEMA

Durante la fase de pruebas del sistema, la Gestión de Configuración del Software será responsable de asegurar la integridad de los datos de los casos de prueba, del producto software y el entorno de utilización, y de cualquier material de pruebas a utilizar. Será responsable de dar al grupo de pruebas todo el material que solicite. En los entornos de prueba automatizados, además se encargará de realizar el control de versiones del material de pruebas.

Cuando la fase de pruebas del sistema se ha terminado, se responsabilizará de archivar el material de pruebas. Añadirá a la documentación de la versión, ya archivada, el informe de los datos de prueba. Los problemas encontrados durante las pruebas serán documentados y guardados en la base de datos de la Cuenta de Estado de Configuración.

PRUEBAS DE ACEPTACIÓN

La Gestión de Configuración del Software controlará totalmente el material disponible para realizar las pruebas de aceptación. Este material será devuelto a la Gestión de Configuración del Software al finalizar las pruebas de aceptación.

El informe de los datos de prueba de aceptación será añadido a la documentación de la versión. Los problemas que se encuentren durante esta fase serán documentados y guardados en la base de datos de la Cuenta de Estado de Configuración.

Se realizará una revisión con todos los resultados de las pruebas, incluyendo las recomendaciones de cada uno de los grupos en el equipo de mantenimiento, para tener una decisión formada sobre la conveniencia de la puesta en producción del sistema. La Gestión de Configuración del Software actualizará la base de datos de la Cuenta de Estado de Configuración con los resultados de las decisiones tomadas durante la revisión, además de dar al jefe de proyecto los informes del estado actual.

PUESTA EN PRODUCCIÓN

Después de ser aprobada por la comisión de revisión final y corroborada por el jefe de proyecto, la Gestión de Configuración del Software será responsable de la distribución del sistema a la comunidad de usuarios. Dependiendo de cómo acceden los usuarios al sistema, la puesta en producción podrá ser llevada a cabo al reemplazar el sistema existente con la nueva versión, o un sistema de copia para usuarios remotos, o la transmisión digital a los usuarios.

Además de la instalación física del sistema, la Gestión de Configuración del Software es responsable de actualizar los registros de configuración para reflejar en ellos la nueva versión, además de archivar el sistema completo, incluyendo todos los datos de la versión paquetizada. La Gestión de Configuración del Software deberá dar copias del sistema para la recuperación posterior del mismo ante cualquier desastre.

CARACTERÍSTICAS DE UNA BUENA INFRAESTRUCTURA PARA EL MANTENIMIENTO DEL SOFTWARE

La corrección de problemas del software en un corto periodo de tiempo y con una alta calidad requiere la realización de diversas prácticas:

- Los problemas puedan ser fácilmente reproducidos – esto necesita una alta calidad de la información sobre el entorno en el que se produjo el problema.
- Los problemas puedan ser comunicados al grupo de desarrollo en un corto periodo de tiempo (minutos u horas).
- Los cambios del código puedan ser relativos a los problemas

- Las pruebas de regresión puedan ser aumentadas con una prueba para garantizar la corrección de los problemas.
- Las nuevas versiones puedan ser liberadas a intervalos controlados.

CAPTURA DEL PROBLEMA

Vamos a describir los procesos y requisitos para garantizar que el grupo de desarrollo pueda realizar de la forma más rápida la resolución de los problemas aparecidos. Uno de los puntos clave es que este grupo actúa como la primera parte del flujo de seguimiento del problema.

La captura del problema es el punto de inicio del proceso de mantenimiento. La dificultad estriba en asegurar que el grupo de soporte y desarrollo reciben la información con la suficiente rapidez y calidad. Para ello se pueden establecer herramientas y procesos para facilitar la captura de información crítica de los problemas surgidos a los usuarios del sistema para que la información pueda estar disponible rápidamente para el grupo de desarrollo.

Probablemente el aspecto más importante para el éxito del mantenimiento de cualquier aplicación es la capacidad el grupo de desarrollo para obtener la información precisa sobre la naturaleza del problema, por lo tanto se debería:

- Capturar automáticamente tanta información como sea posible. Cada uno de los pasos que los usuarios deben realizar para describir los problemas del sistema impedirán su verdadera función. Muchos pasos dan más oportunidad a que la información sea modificada.
- Capturar tanta información de contexto como sea posible. Esto simplifica el hecho de reproducir los procesos para el grupo de desarrollo. Dependiendo del entorno de ejecución, existirán ficheros de la aplicación que resulten cruciales. Por ejemplo, muchas aplicaciones generan ficheros .log que muestran los pasos seguidos hasta el problema.

SEGUIMIENTO DEL PROBLEMA

Una vez los problemas han sido capturados, deberán ser gestionados como recursos dentro del proceso de mantenimiento, como los componentes software son gestionados durante el proceso de desarrollo.

La información recogida en la captura del problema debe ser organizada y gestionada para determinar el estado de las respuestas asociadas al problema. Se requerirá instalar y utilizar un sistema de seguimiento del problema y de las tareas asociadas dentro de las actividades de desarrollo, aseguramiento de calidad, y el grupo de soporte y desarrollo.

Los sistemas de gestión de problemas han realizado durante algún tiempo una función integral dentro del proceso de Mantenimiento del Software. Estos sistemas han dado históricamente una buena infraestructura para la gestión del soporte a las peticiones, requisitos y defectos del software. La información básica sobre un problema tal como el identificador, el origen del problema, la descripción textual, la versión utilizada de la aplicación, el histórico, el estado, los niveles de severidad y los informes, se sabe que están disponibles para los grupos de soporte y desarrollo. La información avanzada también debería ser gestionada.

Las infraestructuras de red y comunicaciones juegan un papel muy importante para poder ser utilizadas para mover la información rápidamente dentro de la organización. La

información de alta calidad sirve de poco si no se comunica al grupo de desarrollo en el menor tiempo posible. De la misma manera, el personal de soporte cercano a los usuarios, debería ser capaz de obtener fácilmente la información sobre el estado de los problemas que han encontrado los usuarios.

Es importante asegurar que existe un mecanismo formal para relacionar los problemas con versiones de la aplicación dentro del entorno de desarrollo.

PRUEBAS DE REGRESIÓN

Mientras que el seguimiento del problema asegura una infraestructura de comunicaciones óptima, las pruebas de regresión nos aseguran una infraestructura dentro del grupo de desarrollo para poder liberar versiones del software de alta calidad.

Tenemos que minimizar los riesgos en el sistema existente para que los cambios o modificaciones realizados para corregir un problema no creen problemas adicionales o tengan efectos laterales no deseados. Para ello se tendrá que adoptar un proceso o utilizar herramientas que ayuden a realizar las pruebas de regresión.

Las pruebas de regresión son particularmente valiosas cuando se utilizan como un repositorio para las pruebas que validan la solución de los defectos del software encontrados. Para cada uno de estos defectos, la prueba de regresión deberá ser aumentada con una prueba que esclarezca que el defecto anterior ha sido corregido, y demostrar que el problema ha desaparecido tras la liberación de la correspondiente versión. Como resultado, la porción de mantenimiento de estas pruebas de regresión, crece con el tiempo. Otros efectos interesantes serán:

- Las pruebas de regresión crecerán en aquellas áreas donde más problemas se han encontrado.
- Las pruebas de regresión garantizarán que los anteriores *bugs* no serán introducidos al sistema como resultado de otros cambios.

CORRECCIONES

Los problemas que se manifiestan como defectos del software requieren especial atención del grupo de desarrollo, y por tanto estos necesitan información adicional sobre el problema.

Para mantener un seguimiento de las relaciones entre las soluciones a los problemas y los cambios del código, las correcciones ofrecen un soporte formal para relacionar uno o más problemas con su conjunto asociado de clases y/o cambios de las versiones.

Una corrección puede ser considerada un componente adicional contenido dentro del problema, o como una entidad independiente que puede ser opcionalmente relacionada con un problema aparecido. La corrección es sólo aplicable si el problema representa un defecto del software. Además, implica la intención de corregir el problema al aplicar un *parche* dentro de la parte afectada de la aplicación, en vez de desarrollar una nueva versión de la misma. En otras palabras, las correcciones individuales serán combinadas y liberadas como un incremento a una versión de la aplicación existente. La noción de corrección implica un cambio que esta localizado en una aplicación software.

PARCHES

Típicamente, las correcciones de defectos del software no se gestionan de manera individual, sino que son manejadas en grupos.

Un parche es un conjunto de correcciones que juntas comprenden un mantenimiento incremental sobre un nivel de la versión de la aplicación. En otras palabra, un parche es una unidad de versión incremental.

La construcción de parches debe ser hecha cuidadosamente para evitar incorporar correcciones conflictivas. El conjunto de correcciones contenidas en un parche y liberadas sobre una versión de la aplicación deben ser en conjunto coherentes y con un buen comportamiento. Las diferentes estrategias que pueden ser utilizadas para construir y gestionar los parches pueden ser, por ejemplo, el parche acumulativo que ofrece un mecanismo para ir de una versión de la aplicación conocida a un mantenimiento correctivo a pesar de otros parches que podrían haber sido liberados.

Cuando los parches alteran la base del software en uso, la lista de parches aplicados deberá ser registrada como información del problema para los que se encuentren en el futuro dentro de la aplicación parcheada.

ANEXO I – INGENIERÍA DEL SOFTWARE

VERIFICACIÓN Y VALIDACIÓN DEL SOFTWARE

Todos los cambios producidos en el software tras la recepción de una Solicitud de Modificación, pueden producir efectos no deseados sobre los requisitos de la aplicación. Aunque durante toda la fase de mantenimiento se hace hincapié sobre este hecho, es cierto que la verificación y validación de que el software cumple con los requisitos establecidos es una de las actividades que, supervisadas por la organización, deben llevarse a cabo para asegurar la calidad del software en fase de mantenimiento.

ASEGURAMIENTO DE LA CALIDAD DEL SOFTWARE

Cuando se realiza cualquier modificación sobre un sistema existente, se deberá considerar el Aseguramiento de la Calidad del Software. Una modificación a un segmento de un sistema puede causar errores que podrían aparecer en algún otro segmento no previsto. Otros elementos a tener en cuenta, además de la modificación del código, serán el control de versiones y la generación de nueva documentación. Para asegurar que se mantiene la calidad para cualquier modificación, los estándar que se deberán serguir son: IEEE 730-1989 y IEEE 983-1986.

Durante el mantenimiento se deberían llevar a cabo los mismos tipos y niveles de aseguramiento de la calidad, por ejemplo inspecciones, revisiones, auditorías, verificaciones, ..., como se realizaron durante la fase de desarrollo. El plan de Mantenimiento del Software es el que especifica el grado y la forma de realizar estas revisiones. Una parte fundamental a cuidar será asegurar que la documentación del sistema original continua describiendo el producto actual.

Durante la utilización del producto, el tiempo disponible para realizar una modificación impide realizar consecuentemente los cambios en la documentación, con la considerable pérdida en el control de la configuración.

GESTIÓN DEL RIESGO

La gestión del riesgo en el Mantenimiento del Software difiere en su mayoría con la gestión del riesgo en el proceso de desarrollo. Las oportunidades de riesgo son más frecuentes, los riesgos vienen de más diversas fuentes, y los proyectos tienen menos libertad para actuar sobre ellos.

Mucha de la literatura sobre la gestión de riesgo pertenece específicamente al desarrollo de nuevo software. El Mantenimiento del Software prácticamente se ha pasado por alto. Esto es comprensible dada la fea imagen que el mantenimiento tiene en la comunidad de ingeniería del software. Por otro lado, las enormes sumas de dinero gastadas en el software hoy día, no van dirigidas al desarrollo, sino a esta parte de la ingeniería del software. Y por una buena razón: no es usual en una aplicación software que sea utilizada durante dos o más décadas.

Corregir este descuido no es tan simple como desplazar el desarrollo. Por supuesto, las dos áreas (desarrollo y mantenimiento) comparten ciertos factores de riesgo, como son la volatilidad de los requisitos, la inexperiencia de los programadores e ingenieros, y las restricciones de tiempo. Pero existen algunas diferencias significativas. La gestión de riesgos en

el mantenimiento generalmente ofrece más oportunidades de aparición de riesgos, además de menos libertad para mitigarlos. El grupo de mantenimiento debe trabajar con un sistema existente con poca o sin documentación. El sistema a menudo tiene varios niveles de cambios desde el diseño original e interfaces con multitud de sistemas que los desarrolladores no esperaban. Cualquier cambio en esta intrincada telaraña puede causar efectos laterales mortales.

La gestión del riesgo en el mantenimiento además supone más atención a los temas relacionados con el cliente. Un sistema en mantenimiento tiene un grupo de usuarios preestablecido que ya trabaja con el sistema sin sobresaltos. En contraste, la evaluación de una nueva versión el grupo de usuarios está empezando a conectar con el sistema, por lo que son más cuidadosos con los riesgos, ya que han ajustado sus expectativas, y tienden a ser más tolerantes con las operaciones erróneas. Por estas razones, un riesgo en mantenimiento implica una consecuencia de riesgo más alta, como la pérdida de una base de usuarios establecida.

VALORACIÓN DEL RIESGO

Las actividades de Mantenimiento del Software consumen considerables recursos para realizar los cambios en el software. Tradicionalmente, los sistemas son probados para detectar los defectos del software. Ya que estos defectos causan fallos con muy distintas consecuencias, la importancia (o riesgo) de cada uno de estos fallos varía.

El riesgo del software es definido como la pérdida potencial debida a un fallo durante un periodo de tiempo específico. El riesgo es medido como un producto entre la frecuencia de las pérdidas y la magnitud o nivel de manifestación de las mismas. La valoración del riesgo empieza con un análisis de la manifestación externa (determinación de la magnitud de las pérdidas que pueden resultar de acciones no válidas). La manifestación externa es localizada dentro del sistema para determinar la magnitud de las pérdidas causadas por fallos en partes del software individuales. La similitud del fallo para cada uno de las partes del software está basado en su uso, verificación, validación, adaptabilidad y tamaño.

En el contexto de mantenimiento, el fallo puede ser orientado al producto u orientado al proceso. Es decir, los fallos del producto y del proceso tienen la capacidad de incrementar los costes del producto y son, por lo tanto, considerados riesgos. Las técnicas de disminución de riesgos en los productos incluyen las métricas de pruebas y mantenibilidad. Las técnicas de disminución de riesgos en los procesos incluyen análisis de impacto de las modificaciones del software.

Para medir el riesgo del software, se pueden realizar varias funciones: identificación de las manifestaciones externas, el análisis de las manifestaciones estructurales y la estimación de la aparición de fallos en el software.

Con la ayuda de una herramienta de ingeniería del software se puede crear una línea de referencia para el proceso de evaluación del riesgo. El conjunto de riesgos que se identifiquen hay que clasificarlos conforme a la severidad percibida, al número de ocurrencias y a la temporalidad de las mismas, utilizando una escala de tres valores, baja, media y alta para la severidad, una escala de probable, no probable y cierto para el número de ocurrencias, y una de inmediato, pronto y tarde.

Cada uno de los factores de escala tienen una definición estándar para ayudar a la consistencia en las estimaciones del mantenimiento. Esta información recoge un valor esperado, que permitirá al grupo de mantenimiento clasificar los riesgos de más significativo a menos significativo. Los miembros del grupo y el jefe de proyecto tras revisar la lista de todos los riesgos identificados y priorizados, seleccionará los primeros para proceder a su mitigación. El número de riesgos a mitigar en cada iteración dependerá de los recursos disponibles, aunque esto en si mismo constituye un riesgo estratégico.

IDENTIFICACION DE LAS MANIFESTACIONES EXTERNAS

La función de identificación de manifestaciones externas tiene dos objetivos primordiales. El primero es determinar que acciones en el entorno del software pueden contribuir a los fallos. El segundo objetivo es valorar la importancia de los fallos.

Para contribuir a estos objetivos, el procedimiento envuelve los siguientes pasos:

a) *Definicitón de los peligros del entorno.* Determinar como se propone la organización utilizar el sistema software, y los peligros potenciales tales como problemas financieros, explosiones, tratamiento incorrecto de pacientes, desinformación, pérdida de vidas, y accidentes.
b) *Identificación de la secuencia de accidentes.* Investigar como pueden ocurrir los accidentes y registrarlos en árboles de eventos, escenarios o diagramas de secuencia de eventos.
c) *Análisis de los modelos de fallos.* Identificar los modelos de fallos de la secuencia de accidentes y registrarlos en árboles de fallos. Las áreas de fallos clave serán identificadas mediante los árboles de fallo.
d) *Análisis de las consecuencias.* Determinar las consecuencias de los fallos por sopesar las pérdidas estimadas para cada uno de los escenarios de accidente. Ya que esto tiene un amplio rango de factores y condiciones, hay que centrarse en los fallos clave.

ANÁLISIS DE LAS MANIFESTACIONES ESTRUCTURALES

El análisis de las manifestaciones estructurales se realiza para descubrir como y donde los fallos del software pueden contribuir a las pérdidas identificadas en la valoración de manifestaciones externas. El objetivo de esta función es asignar niveles de manifestación a los módulos de software individuales basados en su capacidad para causar fallos. El procedimiento incluye las siguientes actividades:

a) *Identificar los modelos de fallo del software.* Indicar donde la información errónea puede contribuir a la pérdida e investigar como puede acabar fallando el software.
b) *Determinar el potencial de fallos del módulo.* Localizar los fallos potenciales relacionados con el modelo de fallos e identificar las relaciones asociadas entre fallos y pérdidas.
c) *Analizar los modelos de utilización.* Localizar donde se utilizan los módulos que potencialmente fallan.
d) *Calcular las manifestaciones del módulo.* Estimar las manifestaciones del módulo sumando sus pérdidas potenciales en todos los escenarios de accidente con los que está relacionado.

APARICIÓN DE FALLOS DEL SOFTWARE

El objetivo de la función de aparición de fallos del software es predecir la aparición de fallos a través de las características del proceso de mantenimiento. Como los procedimientos de pruebas del software, la información acerca de los procesos de pruebas se utiliza para actualizar o confirmar las estimaciones iniciales de la aparición de fallos. La aparición de fallos en el software depende del número de fallos en un módulo y la probabilidad de que el fallo será encontrado durante la operativa del sistema. Es decir, la aparición de fallos depende del número de fallos y de la probabilidad de que los fallos causen más fallos. Los riesgos para cada uno de

los módulos está determinado por la probabilidad de cada uno de los tipos de fallos, de su aparición y del coste del fallo.

MITIGACIÓN DEL RIESGO

Seleccionados ya los riesgos a mitigar, se ideará un plan de mitigación para cada uno de ellos. Dentro del plan se listarán las acciones necesarias y sus fechas de finalización, así como un plan de contingencias y la persona responsable de asegurar que el plan de mitigación se ejecute correctamente. Además se generará una lista de seguimiento del estado del riesgo con la lista de riesgos y sus planes de mitigación, que estará disponible para todos los miembros del grupo de mantenimiento para poder ser revisada periódicamente. La persona responsable actualizará la lista semanalmente. Será necesario hacer una reevaluación de los riesgos cada mes o cada dos meses, normalmente en paralelo con el cumplimiento de los principales hitos del proyecto.

TERMINOLOGÍA: UNA BASE PARA EL CONSENSO

En la mayoría de las organizaciones las palabras ***riesgo, problema, suceso, preocupación, resultado***, tienden a ser utilizadas indistintamente. Esto incrementa la facilidad de desentendimientos. Las organizaciones deberían cuidar de, al menos, utilizar ***riesgo*** y ***problema*** de forma precisa, y permitir que otros sepan que es lo que se entiende por ***resultado***, ***suceso*** y ***preocupación***.

Por ejemplo, ***riesgo*** es un evento con la apariencia de ocurrencia y alguna consecuencia potencialmente negativa. Más simple aún, riesgo es un problema potencial; un ***problema*** es un riesgo cuya apariencia ha alcanzado el cien por cien. Es una certeza.

Es mucho más difícil hacer entender que significa ***suceso***, ***preocupación***, o ***resultado***. La palabra ***resultado*** se puede asociar a un evento con una apariencia y consecuencia más grande que ***suceso*** y ***preocupación***. En cambio, ***suceso*** y ***preocupación*** conlleva más riesgo que ***resultado***.

Por lo tanto, no se ha de intentar definir y forzar una terminología estándar, sino que ha de procurarse que la gente sólo utilice las palabras ***riesgo*** y ***problema*** en sus discusiones. Además será necesario especificar el significado explícito de la palabra ***riesgo*** con los adjetivos: posible, probable, esperado y dudoso. Pero puede ocurrir que, de igual manera, estas palabras se utilicen indistintamente e inconsistentemente para indicar una aparición de riesgo o una consecuencia. Por lo tanto, se hace necesaria la formación del grupo de trabajo para la utilización correcta de estos términos, de tal forma que se pueda describir coherentemente un riesgo en términos de su aparición, consecuencia y temporalidad, además de incluir el grado de confianza de cada uno de ellos.

Otra parte importante de la definición terminológica y el consenso de su utilización son los indicadores de éxito o fracaso en la gestión del riesgo. Es muy difícil demostrar que la gestión del riesgo tiene éxito en el sistema, debido a que la no aparición de un riesgo puede tener su origen en la suerte o en la buena gestión del riesgo.

Los indicadores ayudarán a demostrar que la gestión de riesgo tiene un efecto positivo en el proyecto, y podrán servir además como mecanismo de retroalimentación para indicar el progreso que se realiza al introducir la gestión de riesgo en el proyecto. Se pueden definir tres categorías de indicadores:

Indicadores de Conformidad

Se utiliza para demostrar que el personal ha recibido la formación requerida antes de intentar aplicar al proceso, además de indicar si la gestión del riesgo está siendo aplicada cuando la política de la organización lo requiere.

Se puede utilizar también para descubrir rápidamente si la dificultad encontrada fue causada por el proceso en sí mismo o por un error de formación, así como para medir el nivel global de institucionalización de la gestión del riesgo (la gestión de riesgo está afectada en mayor o menor medida por la organización).

Un indicador de conformidad puede ser el número de personas recibiendo formación general y/o especializada.

Indicador de Efectividad

Se utiliza para tener un sentido de la minuciosidad del proceso de gestión del riesgo, y si está afectado por las decisiones que se toman, además de ver como los riesgos se comunican y gestionan.

Un indicador de efectividad es el número de reuniones no planeadas.

Indicador de Valor

Dando todos los recursos que puedan ser utilizados en la gestión de riesgo, ¿verían el grupo de mantenimiento, la organización y los clientes una notable diferencia?

Para una persona del grupo de trabajo, la diferencia puede ser que tendría más tiempo o información para hacer un trabajo de calidad. Para la organización, puede ser que respondan a más peticiones de usuarios o hagan las planificaciones con mayor calidad. Para el cliente, podría ser alcanzar lo que se le prometio.

Los indicadores de valor son medidos al realizar encuestas periódicas al grupo de trabajo y los clientes respecto a como su percepción de la gestión del riesgo afecta positiva o negativamente al software sobre el que trabajan. Esta tendencia es valorada y comparada con otras tendencias tales como el número de problemas detectados y las solicitudes del cliente completadas. El indicador de valor es el más importante de los tres indicadores ya que son la evidencia más directa de que la organización está aceptando o no la gestión del riesgo.

ROBUSTEZ

La robustez es la capacidad de un sistema de evitar comportamientos catastróficos. Los requerimientos de la robustez pueden identificar funciones críticas cuyo fallo puede ser peligroso para los usuarios o las propiedades. El resultado del procedimiento descrito en la *Identificación de Manifestaciones Externas* podrían ser aplicables. El borrador del estandar P1228 contiene información que puede ser utilizada para crear y mantener un sistema.

SEGURIDAD

El grado por el que el sistema y el acceso a la información necesitan ser protegidos puede tener efecto sobre la manera en la que el sistema es mantenido. Un sistema es seguro si el personal no autorizado no puede conseguir acceso a la información protegida ni al sistema.

La seguridad durante el proceso de mantenimiento debería asegurar los siguiente:

a) La integridad del sistema es preservada al asegurar que sólo personal autorizado tiene acceso al sistema y solo los cambios autorizados son implementados. Esto es realizado en cooperación con las funciones de Gestión de Configuración del Software y del Aseguramiento de la Calidad del Software.
b) Las características de seguridad implementadas durante el desarrollo del sistema no están comprometidas, ni por acciones inadvertidas ni por fallos que cumplen con los requisitos de seguridad existentes.
c) Las nuevas funciones añadidas al sistema son compatibles con las características de seguridad existentes.

MÉTRICAS Y MEDIDAS

El establecimiento y la implementación de un plan de métricas es crítico para la provisión de conocimiento respecto a los niveles de productividad de una organización, también como a la calidad del software mantenido por esta organización. Los estándares IEEE 982.1-1988 y 982.2-1988 dan guías adicionales mediante definiciones, metodologías y razones para implementar un plan de métricas. Las métricas o medidas capturadas para el mantenimiento deberían capacitar a los directores para gestionar los procesos y a los desarrolladores para implementar los procesos.

Para iniciar un proceso de métrica, la jefatura necesita identificar los factores técnicos que reflejan la calidad técnica y de gestión del software que está siendo mantenido. Una vez estos factores son identificados con indicadores, entonces las medidas deberían ser desarrolladas con lo que corresponde a factores técnicos y cuantificar estos factores. Se sugiere que la selección de los factores técnicos y sus correspondientes métricas sean optimizadas de forma que solo los factores que son pertinentes a la fase específica del proceso de mantenimiento son realizados durante su respectiva fase del proceso.

Una vez se han identificado, se deberá realizar un análisis del coste-beneficio para determinar el mejor valor que puede ser alcanzado por la organización (en términos de aumento de la productividad y sobre todo mejora de la gestión del proceso), a cambio del esfuerzo consumido en recoger y analizar los datos de medida. Al menos, el esfuerzo en términos de horas de trabajo deberá ser recogido y convertido en coste, utilizando la relación de trabajo interno de la organización. Adicionalmente, se recogerán algunas medidas de la funcionalidad, además de los errores generados y clasificados por prioridad y tipo.

Algunas de las medidas comunes de la funcionalidad son: líneas de código fuente, puntos de función y puntos característicos. Se elija el método que se elija, debería estar bien definido y en acuerdo con la organización. Las herramientas utilizadas para la recogida de medidas, el análisis, y la valoración deberían ser validadas, calibradas y utilizadas por la propia organización.

En el mundo del mantenimiento del software, existen tres procesos de mayor coste en su realización: documentación, comunicación/coordinación y pruebas. Por lo tanto, cualquier plan de métrica en el mantenimiento del software deberá incluir medidas que indiquen minuciosamente el coste de estos procesos, tales como páginas modificadas de documentación, esfuerzo en la negociación del alcance del trabajo que será incluido en los cambios del software, y clasificación de los errores por prioridad y tipo.

Un perfil de complejidad de cada uno de los programas estará compuesto, aunque no limitado a lo siguiente:

a) Tamaño del programa (número de sentencias o instrucciones)
b) Número de módulos del programa
c) Número de variables del programa

d) Número de variables globales del programa
e) Tamaño medio del módulo (en sentencias)
f) Número medio de comparaciones por módulo
g) Número medio de módulos que acceden a variables globales
h) Lista de módulos comunes.
i) Lista de módulos que acceden a más de la media del número de variables globales
j) Lista de módulos que exceden el límite del tamaño del módulo de 50 sentencias o excede el límite de comparaciones por módulo de 10 comparaciones.

La principal fuente de datos para la toma de medidas de la aplicación es la librería de configuración del software. Nada más capturar los primeros datos, debería crearse un repositorio para almacenar esta información y las siguientes medidas que se recojan. Este repositorio deberá ser directamente relacionado con el sistema de control de modificaciones de la organización. Este sistema es la principal fuente de datos para la actualización constante del inventario de perfiles de solicitudes.

POLÍTICA DE REEMPLAZAMIENTO DEL SOFTWARE

La planificación del mantenimiento deberá ser considerada para todos los sistemas, incluso aquellos bajo su desarrollo inicial. Aunque todos los sistemas necesitan mantenimiento, llega un momento en el que el mantenimiento no es técnica o físicamente posible. La negociación de los recursos, fondos, prioridades, etc, podrá indicar que un sistema deberá ser reemplazado antes que modificado. La política de gestión que puede ayudar a tomar una decisión correcta incluye la determinación de los siguientes valores:

a) Obsolescencia del sistema o relación de fallos.
b) Edad del código superior a N años.
c) Complejidad de la estructura o lógica del sistema
d) Nuevo hardware
e) Requisitos de recursos excesivos
f) Inexistente o deficiente documentación o especificaciones de diseño.

TECNOLOGÍAS DE SOPORTE AL MANTENIMIENTO DEL SOFTWARE

REINGENIERÍA

El Mantenimiento del Software es una parte significativa del proceso de ingeniería del software. El nuevo código, inevitablemente requiere cambios: nuevas regulaciones, cambios en funciones o las reglas que componen estas funciones, correcciones a problemas, extensiones a funciones, etc. Estos cambios son tratados normalmente como una parte menor del proceso de desarrollo y delegado a programadores con menor experiencia. Por lo general, se asume que este sistema de nuevo desarrollo tendrá una vida corta y será reconstruido una vez que los cambios que necesite lleguen a ser demasiado costosos de abarcar. Sin embargo, estos sistema continuan siendo de gran valor para la organización, y por lo tanto, la revitalización de estos viejos sistemas llegan a ser una opción práctica.

Como un subconjunto del Mantenimiento del Software, recientemente la reingeniería ha recibido una atención significativa. El redesarrollo de los sistemas clave llega a ser extremadamente costoso, tanto en dinero como en trastornos. Un análisis crítico de la documentación del software y la reingeniería selectiva es una forma más evolucionaría de actualizar los viejos sistemas a los estándares actuales y a las nuevas tecnologías. La reingeniería no será sólo revitalizar el sistema, sino proveer material reutilizable para futuros

sistemas y crear el marco de trabajo funcional para un entorno orientado a objetos. Las técnicas para llevarlo a cabo han estado siempre en otras disciplinas de la ingeniería. Sin embargo, su aplicación al software es muy reciente, y consecuentemente, las herramientas para soportar la reingeniería están apareciendo.

La reingeniería como aproximación está compuesta generalmente de dos componentes: ingeniería inversa e ingeniería 'directa'. La ingeniería inversa no cambia el sistema. Proporciona una vista alternativa del sistema en un nivel de abstracción diferente. Esto significa la redocumentación del código fuente como esquemas, diagramas estructurados, o diagramas de flujo para entender la lógica del código. La ingeniería 'directa' es el proceso de construcción de un sistema. Este proceso comienza con una estructura del sistema ya existente, que es el marco de trabajo para cambios y mejoras.

INGENIERÍA INVERSA

Muchos sistemas tienen el código fuente como la única representación fiable. Esto es más común en sistemas muy antiguos que han tenido muchos cambios durante su ciclo de vida. Estos sistemas tienen un crecimiento substancial desde su inicio, y generalmente no han tenido actualizaciones de su documentación.

Aquí es donde la ingeniería inversa es la técnica recomendada para redocumentar el sistema.

El primer documento que se necesita para navegar fácilmente en el sistema y encontrar la localización de un problema a nivel de análisis del Mantenimiento del Sistema es el esquema del programa.

Los pasos a seguir para realizar la ingeniería inversa son:

a) Disección del código fuente en unidades formales.
b) Descripción semántica de las unidades formales y declaración de unidades funcionales.
c) Creación de esquemas de entrada/salida de unidades.
d) Declaración y descripción semántica de circuitos lineales.
e) Declaración y descripción semántica de aplicaciones del sistema.
f) Creación de la anatomía del sistema.

COMPRENSION DE LOS PROGRAMAS

La comprensión de los programas en uno de los principales factores para alcanzar un Mantenimiento del Software efectivo y ser capaz de evolucionar con éxito los sistemas software. Durante años, los investigadores han intentado entender como los programadores comprendían los programas durante la evolución y mantenimiento del software. Generalmente se asocian cinco tipos de tareas con la evolución y Mantenimiento del Software: Mantenimiento Adaptativo, Perfectivo, y Correctivo; Reutilización; y Influencia del Código. Cada una de estos tipos de tareas recoge ciertas actividades:

TAREA DE MANTENIMIENTO	ACTIVIDADES
Mantenimiento Adaptativo.	Entender el sistema.
	Definir requisitos de adaptación.
	Desarrollar el diseño de adaptación preliminar y detallado.

	Cambiar el código
	Depurar
	Realizar pruebas de regresión
Mantenimiento Perfectivo	Entender el sistema
	Definición de requisitos para mejorarlo
	Desarrollo del diseño de perfeccionamiento
	Cambiar/Añadir código
	Depurar
	Realizar pruebas de regresión
Mantenimiento Correctivo	Entender el sistema
	Generar/Evaluar hipótesis sobre el problema
	Reparar el código
	Realizar pruebas de regresión
Reutilización	Entender el problema, encontrar la solución basándose en componentes reutilizables
	Localizar componentes
	Integrar componentes
Influencia del Código	Entender el problema, encontrar la solución basándose en componentes predefinidos
	Reconfigurar la solución para aumentar la utilización de componentes predefinidos
	Obtener y modificar componentes predefinidos
	Integrar componentes modificados

Algunas actividades, tales como *Entender el sistema o el problema*, son comunes a varias tareas. Para analizar los procesos cognoscitivos tras estas tareas, los investigadores han desarrollado varios modelos.

A causa del limitado conocimiento sobre el conocimiento especializado necesario para algunas tareas de mantenimiento, los modelos de cognición del código no representan cada una de estas tareas. La mayoría de los modelos asume que el objetivo es entender todo el código, preferiblemente a una sola parte en particular, por ejemplo la depuración. Mientras estos modelos generales pueden fomentar un completo entendimiento de una pieza de código, no siempre se pueden aplicar a tareas especializadas con las que se emplearían otras técnicas.

Otras cuestiones por resolver serán, por ejemplo, considerando de forma particular el mantenimiento y evolución de código a gran escala. Estas cuestiones son relativas a la escalabilidad de resultados experimentales existentes con pequeños programas, la validez y credibilidad de los resultados basados en procedimientos experimentales, y la disponibilidad de los datos.

El proceso de comprensión de un programa utiliza el conocimiento existente para adquirir nuevos conocimientos que al final alcanza el objetivo de la tarea de comprensión del código. Este proceso está referido tanto al conocimiento existente como al adquirido para construir un modelo mental del software en consideración. La comprensión depende de las estrategias a seguir. Aunque estas estrategias de cognición varían, todas siguen los siguientes pasos: formulan hipótesis y después las resuelven, las revisan o las abandonan.

Los programadores poseen dos tipos de conocimiento: conocimiento general, que es independiente del software específico que están intentando entender; conocimiento específico del software, que representa su nivel de entendimiento de la aplicación software. Durante el proceso de entendimiento, los programadores adquieren más conocimiento específico del software, pero puede necesitar más conocimiento general (por ejemplo como trabaja un algoritmo round-robin). El conocimiento que posee el programador está relacionado con los

lenguajes de programación y el entorno operativo, los principios de programación, la arquitectura utilizada, los algoritmos, y otras posibles soluciones. Si el programador ha trabajado antes con el código, el conocimiento que tenga incluye además cualquier modelo mental del software que posea.

El nuevo conocimiento a adquirir, principalmente está relacionado con el producto software. Este se adquiere a través del proceso de entendimiento del código. Este conocimiento está relacionado con la funcionalidad, la arquitectura del software, la forma en que los algoritmos y objetos son implementados, los métodos de control, de flujo de datos, etc. Obviamente esto alberga muchos niveles de abstracción desde *"esto es un sistema operativo"* hasta *"la variable X es incrementada en el bucle"*. El proceso de entendimiento enlaza el conocimiento que posee el programador con el conocimiento del software hasta que el programador cree que entiende el código. Este conjunto de pares es un modelo mental, que puede ser completo o incompleto.

SISTEMAS LEGADOS

Los sistemas legados o *legacy systems* están caracterizados por una o más de las siguientes propiedades: fueron implementados hace muchos años, su tecnología está obsoleta, su estructura está deteriorada, representan una gran inversión, contienen reglas de negocio ya no necesarias o inservibles, no pueden ser reemplazados fácilmente, y/o los autores originales no están disponibles. Estas propiedades son las razones por las que trabajar con estos sistemas es muy complicado. El problema tan actual del año 2000 es uno de los principales síntomas de esta dificultad. Cuando se trabaja con estos sistemas, el proceso de comprensión consume la mayoría de los recursos, o por lo menos una mayor proporción de estos.

Otro proceso importante que dificulta el trabajo con los sistemas legados incluye el análisis de impacto de los cambios, su localización y su propagación. Además se deberá realizar una reingeniería que incluya la reestructuración y la redocumentación.

MANTENIMIENTO EN LA ORIENTACION A OBJETOS

REPRESENTACIÓN DE LAS DEPENDENCIAS

El Mantenimiento del Software es un proceso muy costoso debido a que cada modificación de un programa debe tener en cuenta muchas dependencias o relaciones complejas en el software existente. Un conocimiento de las dependencias del programa es, por lo tanto, una paso inevitable en la realización de modificaciones eficientes del software.

Las dependencias de un programa son relaciones de dependencia mantenidas entre las sentencias del código, que están determinadas por los flujos de control y los flujos de datos del programa. Intuitivamente, si la ejecución de una sentencia directamente o indirectamente afecta a la ejecución de otra en un programa, ahí puede existir alguna dependencia entre las sentencias.

Muchas de las optimizaciones de los compiladores dependen a su vez del análisis de las dependencias de un programa, y que por lo general se representan con la forma de un PDG (*Program Dependence Graph*). Este gráfico, aunque fue propuesto originalmente para las optimizaciones de los compiladores, ha sido aplicado en varias actividades de ingeniería del software, como son: modularización , depuración, pruebas, mantenimiento y métricas de complejidad.

Aunque se han propuesto un gran número de representaciones basadas en las dependencias, la mayoría corresponden a la programación estructurada, y se le ha prestado poca atención a la programación orientada a objetos.

Comprender los programas orientados a objetos concurrentes es mucho más difícil que entender la programación orientada a objetos secuencial. En los concurrentes, además de las características de la orientación a objetos tales como las clases y sus instancias, la herencia, el polimorfismo, existen además elementos de concurrencia tales como sincronización entre procesos y comunicaciones.

Una de estas representaciones es la SDN (*System Dependence Net*), que consiste de una colección de gráficos de dependencias en el que cada uno representa un procedimiento principal, un procedimiento aislado, o un método de una clase en el programa. Además tiene arcos para representar las dependencias directas entre una llamada y el procedimiento/método llamado, así como las dependencias de datos transmitidos entre procedimientos.

APLICACIONES DE LAS REPRESENTACIONES DE DEPENDENCIAS

Si tenemos un gráfico SDN como nuestra representación de dependencias en un programa orientado a objetos, se pueden discutir algunas de las más importantes aplicaciones del SDN en un entorno de mantenimiento para el software orientado a objetos.

Modularización

La aplicación más directa de un diagrama SDN es la posibilidad de modularizar los programas orientados a objetos. La modularización de un programa es una técnica de descomposición por la que se extraen las sentencias relacionadas con un cálculo particular. Un módulo estará compuesto por aquellas partes de un programa que pueden afectar directa o indirectamente a los valores calculados en algún punto de interés del programa, al que denominaremos criterio de modularización. Esta técnica tiene muchas aplicaciones en las actividades de ingeniería del software, tales como: comprensión del programa, depuración, pruebas, mantenimiento, ingeniería inversa, ...

Mantenimiento Del Software

Uno de los problemas en el Mantenimiento del Software es el efecto de ola, por ejemplo, si un código cambia en un programa, afectará al comportamiento de otros códigos en el mismo programa. Para evitar este problema, es necesario conocer que variables y que sentencias se verán afectadas por la modificación de una variable, y también, que variables en que sentencias afectarán a una variable modificada durante el mantenimiento de un programa orientado a objetos. Para evitarlo se puede utilizar la modularización del código.

Métricas De Complejidad Del Software

Las métricas del software tienen multitud de aplicaciones en las actividades de ingeniería del software. Debido a que el diagrama SDN de un programa orientado a objetos puede ser utilizado para representar las propiedades de los flujos de datos en el programa, basándose en el diagrama SDN se pueden definir algunas métricas para medir la complejidad de los programas orientados a objetos desde diversos puntos de vista. Por ejemplo, la métrica definida por la suma de todas la dependencias entre las sentencias de un programa, puede ser utilizada para medir la complejidad total del programa, y la proporción entre las dependencias relativas a la concurrencia y el total de las dependencias en el programa puede ser utilizada para medir el grado de concurrencia del mismo.

SOFTWARE BASADO EN COMPONENTES

La tendencia actual indica que los sistemas están incrementando su arquitectura basada en componentes. Las técnicas emergentes de componentes independientes y vendibles por separado y la reutilización de componentes están alterando sin duda alguna el Mantenimiento del Software. Mientras el mantenimiento tradicional era, en gran parte, un microcosmos de desarrollos continuados, el software basado en componentes tiene un ciclo de vida tanto para el fabricante como para las organizaciones que lo compran para integrarlo en el desarrollo de sus aplicaciones. Estos componentes incrementan las incertidumbres sobre el mantenimiento de las aplicaciones ya que ofrecen una cara oculta e incontrolada, y los cambios sobre el componente pueden también afectar al comportamiento de la aplicación.

MANTENIMIENTO EN EL FUTURO

Cuando el ancho de banda de las redes y las herramientas de replicación y transferencia de ficheros sean más potentes, las estrategias y las elecciones para la puesta en producción de un software aumentarán. En el pasado, instalar una imagen completa del software cada vez que se necesitaba hacer un mantenimiento correctivo no era aceptable. Al ser el desarrollo orientado a objetos basado en componentes, ofrece la flexibilidad de liberar aplicaciones de muchas maneras. Sin embargo, la complejidad de realizar el seguimiento de las versiones de los componentes que se instalan y en que cliente, lo hacen más desalentador. Cuando el tamaño de un componente individual decrece, y el número de componentes aumenta, se complica la gestión de las relaciones dentro del contexto de la aplicación. Se hace más grande la necesidad de utilizar buenas estrategias heurísticas de instalación.

ANEXO II

RELACIÓN ENTRE LOS ESTANDARES IEEE Y EL IEEE 1219-1992

A continuación se incluye una tabla con la relación entre las distintas fases y etapas que se siguen en el estándar IEEE 1219-1992 que nos ocupa, y el resto de los estándares de ingeniería del software que hay que tener en cuenta para asegurar la calidad de su realización.

FASE	ETAPA	ESTANDARES IEEE
Identificación y clasificación	Proceso	
	Control	
Análisis	Proceso	830-1984 y 1074-1991
	Control	830-1984
Diseño	Proceso	830-1984, 1016-1987 y 1074-1991
	Control	830-1984 y 1016-1987
Implementación	Proceso	1008-1987 y 1074-1991
	Control	829-1983 y 1008-1987
Pruebas del Sistema	Proceso	829-1983, 1074-1991, 1028-1988 y 1012-1986
	Control	829-1983, 1028-1988 y 1012-1986
Pruebas de Aceptación	Proceso	1074-1991 y 1012-1986
	Control	829-1983, 1028-1988 y 1012-1986
Puesta en Producción	Proceso	
	Control	1063-1987

RELACIÓN DE LOS PROCESOS DE INGENIERÍA Y LOS ESTÁNDAR IEEE QUE LOS DEFINE.

A continuación se incluye una tabla con la relación entre los distintos procesos de ingeniería del software que se han mencionado a lo largo del estándar IEEE 1219, y los diversos estándares de ingeniería del software que hay que tener en cuenta para asegurar la calidad de su realización.

PROCESO	ESTANDARES IEEE
Gestión de Configuración	828-1990 y 1042-1987
Métricas y medidas	982.1-1988 y 982.2-1988
Planificación	829-1983, 1012-1986 y 1058-1987
Herramientas y Técnicas	
Aseguramiento de la Calidad	730-1989 y 983-1992
Valoración del Riesgo	730-1989 y 982.2-1988
Robustez	
Seguridad	

FORMULARIOS Y PLANTILLAS UTILIZADAS EN EL ESTÁNDAR

Durante el proceso de Mantenimiento del Software, y como se ha dejado ver a lo largo del estándar IEEE 1219-1992, para poder recabar la información necesaria tanto de la jefatura e ingenieros del software del proyecto, como de la propia organización, es obligado tener unas plantillas y formularios bien definidos y estandarizados. A continuación se da una lista de los formularios que se podrían utilizar y el estándar que los define:

1. Log de pruebas – IEEE 829-1983
2. Informe de incidencias de pruebas – IEEE 829-1983
3. Informe de resumen de pruebas – IEEE 829-1983
4. Especificación del diseño de pruebas – IEEE 829-1983
5. Solicitud de Cambio del Software/Sistema – IEEE 1042-1987
6. Autorización de Cambio del Software – IEEE 1042-1987

BIBLIOGRAFÍA

Agramal, H., Alberi, J., Horgan, J., Li, J., London, S., Wong, E., Ghosh, S., Wilde, N., 1998. Mining Systems Test to Aid Software Maintenance, Computer, July 1998

Charette, R., Adams, K., White, M., 1997. Managing Risk in Software Maintenance, IEEE Software, Pags.43-50, Mayo/Junio

Department of Computer Science, Colorado State University, Program Comprehension During Software Maintenance and Evolution. Disponible: http://dlib.computer.org/dynaweb/ (Acceso 1-Marzo-1999)

International Conference on Software Maintenance (ICSM), 1998. Disponible: http://computer.org/conferen/home/icsm (Acceso 26-2-1999)

Lesage, D. Patterns for OO Software Maintenance. Disponible: http://www.unity-software.com

Piattini, M., Villalba, J., Ruiz, F., Fernández, I., Polo, M., Bastanchury, T., Martínez, M.A., 1998. Mantenimiento del Software. Conceptos, métodos, herramientas y outsourcing. Ra-ma editorial, Madrid.

Software Engineering Process Technology (Sept), 1999, Software Maintenance Standards. Disponible: http://www.12207.com/maintena.htm (Acceso 19-Abril-1999)

The Institute of Electrical and Electronics Engineers. Inc. 1993, IEEE Standard for Software Maintenance

Zhao, J., Cheng. J., Ushijima, K. A Dependence-Based Representation for Concurrent Object-Oriented Software Maintenance.