TEMA 4: QUALITY ASSURANCE: CONTROL DE CALIDAD DE PROYECTOS

4.0 Introducción

Una de las causas primordiales por la que una gran cantidad de proyectos, que a priori se vislumbraban con un gran futuro, se hayan visto conducidos al fracaso, y hayan tenido que ser abandonados tras su pase a un estado de producción, e incluso antes, durante las fases de desarrollo, es sin duda la carencia de unos Estándares rigurosos de Control de Calidad.

Según afirma Thomas DeMarco [DeMarco, 1982], "*por término medio, los productos software en los Estados Unidos contienen mas de un defecto por cada mil líneas de código, y resulta tan caro el modificarlos que la mayoría de las veces es mas barato el volverlos a escribir*."

Si consideramos ahora los problemas que se originan al tener que desechar un trabajo realizado a un alto costo por todo un Equipo de Proyecto, y añadimos el tiempo perdido en el desarrollo, y agregamos las demoras en la entrega del producto terminado y sumamos la falta de credibilidad que este tipo de acciones supone, no es difícil comprender por que tantas "casas de software" han tenido que cerrar sus puertas en estos últimos años.

Por ello, muchos creadores de software, tratando de mejorar la situación, Están comenzando a gastar una porción significativa de sus presupuestos en actividades de Control de Calidad, aunque gran parte de esos programas poco han hecho realmente para mejorar la calidad del software, pero han tenido sin duda un gran éxito en obstruir el desarrollo de los nuevos proyectos por crear burocracias que refuercen unos Estándares encorsetados, insistiendo en completar interminables "checklists" y adhiriéndose a metodologías de desarrollo rígidas y obsoletas.

Ante esto cabe preguntarse: ¿Es posible que un programa de Control de Calidad dificulte la creatividad, obstaculice el desarrollo y constituya una pérdida de tiempo y dinero? ¿Acaso el Control de Calidad está destinado a servir tan solo como "gancho" en las ventas del software? ¿Cómo se pueden conjugar las ventajas que un programa de Control de Calidad aporta con los inconvenientes que proporciona?

El objetivo de este tema es intentar contestar a estas y a muchas otras preguntas que se plantean cuando se intenta implantar con éxito un Sistema de Control de Calidad en la Gestión de Proyectos Informáticos.

4.1 Concepto de Control de Calidad

Hasta ahora, hemos mencionado ya varias veces la palabra calidad, citando su importancia en el desarrollo de proyectos. Es este el momento en que deberemos de ocuparnos de definir lo que se entiende por Control de Calidad, su necesidad real y los trabajos que hay que desarrollar para implementar un efectivo control en la calidad de los productos.

No resulta fácil definir la calidad. Es uno de esos muchos conceptos que todo el mundo conoce y entiende sin dificultad, pero que resultan bastante engorrosos de expresar con palabras. Tal vez una definición adecuada seria la adoptada por la empresa Rank Xerox: "*La calidad consiste en hacer las cosas bien a la primera, proporcionándole a los clientes externos e internos, productos y servicios que satisfagan plenamente los requerimientos acordados*".

Si analizamos despacio la definición podremos observar dos puntos a destacar. En primer lugar, se deben conseguir los resultados de calidad apetecidos a la primera. ¿Utopía? Todo depende de como queramos entender la definición. La implantación de un sistema de trabajo con calidad es un proceso largo, normalmente entre 3 ó 4 años, durante los cuales es necesario cambiar el "modus operandi", la forma de hacer las cosas, dentro de la totalidad de la organización. ¡Comenzando por la Dirección!. Luego se ir descendiendo paulatinamente por el árbol jerárquico hasta llegar finalmente a niveles operativos. Ante este lento proceso, la única forma de llegar a buen puerto es manteniendo la idea clave de hacer las cosas bien a la primera, aún a sabiendas de la dificultad (o imposibilidad, si se quiere) que esto supone, pero teniendo siempre presente el objetivo trazado desde un principio: la consecución de un trabajo con cero errores. Y es una larga labor de mentalización a todos los niveles la que nos obliga a recordar que en nuestra organización, las cosas se hacen bien a la primera y que nuestro nivel de errores es cero. Debemos afirmar que en nuestro ámbito somos los mejores, sin ninguna clase de dudas. Si se consigue esa mentalización en todo el ámbito de la empresa, se habrá dado un paso de gigante hacia la consecución de un trabajo de calidad real y hacia la implantación de un sistema de trabajo en un entorno de Control de Calidad.

El segundo punto a destacar en la definición es la consecución de productos que satisfagan los requerimientos acordados. No se trata de que un determinado producto software sea "el mejor", sino que, simplemente, se ajuste a las especificaciones dadas por el Usuario y funcione de acuerdo con ellas. Es necesario por tanto emplear todo el tiempo que sea preciso en el conocimiento y negociación de los requerimientos, puesto que una vez que hayan sido aceptados por ambas partes, estos deberán de ser cumplidos en su totalidad.

Con estas premisas en mente, podemos definir el Control de Calidad como un *"conjunto de actitudes y técnicas conducentes a la obtención de un producto software que pueda ser desarrollado en tiempo y en presupuesto, y que obtenga la aprobación del usuario final"*, ya que la calidad carece de significación si no es percibida por éste.

4.2 Necesidad de Control de Calidad

De todo lo expuesto anteriormente se desprende que la implantación de un Sistema de Control de Calidad es un proceso largo y costoso y que, además, puede no llegar a buen término. Es necesario entonces poder disponer de argumentos válidos, además de los intuitivos, para poder defender la postura de la necesidad de QA ante la dirección de la organización, la cual debe de ser la primera convencida y la primera en actuar, como ya se vio anteriormente.

En un primer momento, una postura de prestigio e imagen podría considerarse un argumento válido a favor de QA. Hoy en día, la palabra calidad vende, y mucha

de la propaganda que nos rodea se basa precisamente en la oferta de productos de calidad. Si profundizamos un poco, lo que subyace tras este argumento es la necesidad de conquistar mantener una determinada posición de mercado, trazando un plan de superación de la competencia, apoyado por un sistema de QA. Pero, ¿cómo puede QA ayudar cuando los costes de su implantación son tan elevados? ¿de qué nos sirve un Control de Calidad si para implantarlo la organización da en quiebra?

Naturalmente, se trata de una exageración. Pero en definitiva, y volviendo al terreno de lo real, en el fondo lo que se esconde es un problema de costes, y el coste de la calidad debe de tener su contrapartida y debe de poder ser amortizable.

Para ello, consideremos el esfuerzo de realización de cambios en un sistema en producción, como se muestra en la figura 4.1. Cualquier cambio, por pequeño que sea, una vez que el sistema está en funcionamiento, representa un esfuerzo muy superior en un sistema sin control de calidad que en un sistema con QA.

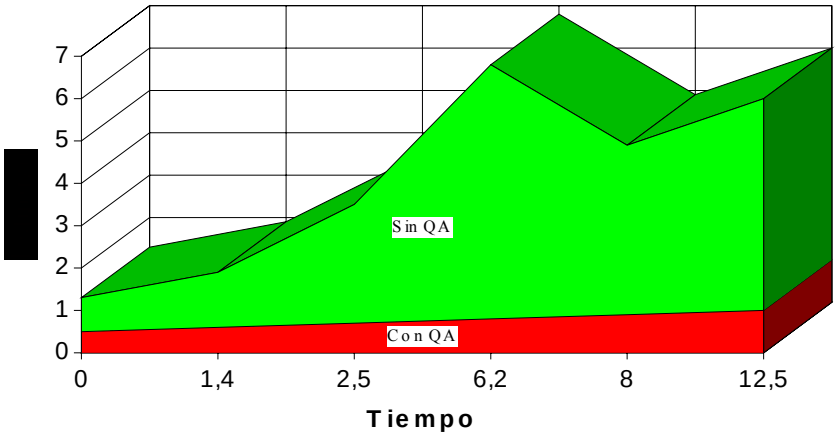

Fig. 4.1

Esfuerzo de cambios de post-implementación

Consideremos las precisiones hechas por Boehm [Boehm, 1980]:

* El porcentaje de los esfuerzos de un departamento de proceso de datos dedicados a tareas de mantenimiento ha estado creciendo firmemente durante 30 años y hoy representa mas del 50% del esfuerzo total

* El coste de detección y corrección de errores crece enormemente durante el ciclo de vida del software. La corrección de errores introducidos en el Análisis y que no son suprimidos hasta la fase de Mantenimiento, cuesta 10 veces mas que si hubieran sido detectados con anterioridad

* El 50% del coste de supresión de errores durante las fases de Pruebas y Mantenimiento puede ser atribuido a la necesidad de corregir defectos introducidos en fases anteriores

Las cifras de Boehm datan de 1980, pero los costes de mantenimiento no han cesado de subir. Hoy en día, se suele estimar que el mantenimiento de Aplicaciones desarrolladas sin Control de Calidad, supone alrededor de un 80% de los recursos de cualquier Organización. Si la utilización de técnicas de Control de Calidad consiguen rebajar tan solo una pequeña parte de estos costes, estará más que de sobra justificada la implantación de Control de Calidad en el desarrollo y gestión de proyectos.

4.3 Requisitos de QA

Antes de adentrarnos en el mundo de QA, pasaremos revista a una serie de requisitos sin los cuales no se debería intentar la implantación de un Sistema de Control de Calidad.

En primer lugar, Control de Calidad deberá de ser el Organismo Supervisor del Desarrollo de Sistemas Informáticos, y depender directamente de la Alta Dirección de la Empresa. Esto es fácilmente entendible si se piensa que una de las labores fundamentales de QA es la detección de errores cometidos durante el desarrollo de las aplicaciones y antes de su puesta en producción, así como la prevención de los mismos. La idea en este punto estriba en que QA realiza mejor su trabajo cuantos mas errores detecta, por lo que la labor de QA, en principio, no es muy bien aceptada por parte del departamento de Desarrollo de Aplicaciones (Desarrollo). Por otra parte, las tareas de QA consumen una parte importante del tiempo total del proyecto, tiempo necesario para pasar todas las revisiones y controles que más adelante se detallarán. Así, cuando los proyectos se retrasan (y se retrasan casi siempre), la parte mas sacrificable parece ser, a priori, la realizada por QA, puesto que QA no aporta código ejecutable. ¿Estamos mal de tiempo? Los plazos deben de cumplirse... Fácil: que QA no pruebe tanto (y en caso extremo, que no pruebe *nada*) Este simple razonamiento se escuchar tantas veces y procedente de personas tan dispares (y tan "altamente cualificadas"), que se podría incluso caer en la tentación de seguirlo. Por tanto, si QA no dispone de la autoridad suficiente que le permita libertad de acción, y que implique norma de obligado cumplimiento, su labor se verá indefectiblemente recortada e incluso anulada por completo en algunos casos.

Además, y por razones obvias, QA deberá de ser una sección (o departamento) totalmente independiente de Desarrollo, para que su labor pueda ser eficaz, funcionando normalmente como staff de la alta dirección de la empresa.

Asimismo, deberán de existir en la organización unos Estándares de análisis y programación perfectamente desarrollados y en funcionamiento, una buena metodología de análisis y desarrollo de aplicaciones y unas normas de QA publicadas y aceptadas por todos, ya que Desarrollo debe de saber en todo momento cual es el rasero por el que se le va a medir.

Si añadimos a todo esto la mentalidad antes comentada de "ser los mejores" y trabajar con "cero errores", empezando por la dirección y terminando con el último mono de la empresa, tendremos el caldo de cultivo adecuado para una buena implantación de un Sistema de Control de Calidad que funcione.

4.4 Ambito y Severidad de QA

El control de calidad *real* se consigue desarrollando métodos fiables para detectar y eliminar defectos en las fases tempranas del desarrollo, y midiendo la calidad actual de los productos terminados, por lo tanto el Control de Calidad abarcar todo el Ciclo de Vida del Proyecto, todo el camino desde las Especificaciones de los Requerimientos, pasando por el diseño de la Base de Datos, el Desarrollo de Aplicaciones, la Migración a Producción, la Operación y el Mantenimiento.

La severidad con que se aplicará el control de calidad depender en gran parte de lo crítico de la aplicación, el tamaño del proyecto, la fase que se esté auditando, etc., debiendo de desarrollarse un baremo de severidad, otorgando una puntuación que permita evaluar cada caso concreto. No es lo mismo un proyecto "para tirar", para andar por casa, que un proyecto que va a hacer de nuestra empresa la mas envidiada del mercado; ni es lo mismo un proyecto de 5.000 líneas de código fuente que otro de 50.000. El sistema de QA deberá adaptar su severidad en cada caso, so pena de estar condenado al fracaso a muy corto plazo. De no hacerlo así, se caería en el riesgo mencionado al comienzo de este tema: un Control de Calidad rígido que frenaría en gran manera la labor de desarrollo de aplicaciones.

Cada instalación deberá de desarrollar sus propias normas y sus propios niveles de severidad, por lo que no es posible dar valores exactos, ya que varían en cada caso. No obstante, y a título de orientación, unos valores adecuados, por ejemplo, para el desarrollo de programas podrían ser los siguientes:

El programa realiza consultas	+	1 punto
El programa realiza altas	+	2 puntos
El programa realiza modificaciones	+	2 puntos
El programa realiza bajas	+	3 puntos
Por cada Entidad manejada	+	0.25 puntos
Etc.		

El total de puntos nos dar la severidad de QA, de acuerdo con la cual se podrán establecer los tiempos mínimos necesarios de prueba, por ejemplo:

Severidad	Horas/hombre
1 a 3	3
4 a 6	5
7 a 8	9
9 a 10	15

Y así sucesivamente. De esta manera se tendrá una buena estimación del coste (en horas/hombre, meses/hombre) de Control de Calidad.

Por otra parte, serán objeto de QA no solo los programas, sino todas las categorías de datos posibles en un Diccionario de Datos, es decir, Esquemas, Subesquemas, Registros, Informes, Documentos "fuente", Transacciones, etc., etc. Asimismo se controlará la calidad del Software de Sistemas y Aplicaciones y el Hardware en lo tocante a pantallas y terminales, entre otros.

4.5 Niveles de QA

Dentro del entorno de QA se deben de establecer determinados niveles, por los que discurrirán los proyectos, y que deberán de ir superando hasta alcanzar el alto grado de fiabilidad necesario.

En un entorno normal de operación, consideraremos hasta cuatro niveles diferentes: QA de Desarrollo, QA Interno, QA Independiente y QA de Migración, aunque en algunos casos se pueda prescindir de QA Interno, sobre todo en proyectos pequeños, en organizaciones pequeñas, y sus funciones sean asumidas por QA Independiente.

4.5.1 QA de Desarrollo

Es el control de calidad realizado por los propios programadores al finalizar un trabajo. Es un primer paso para verificar que lo que se ha desarrollado funciona y es operativo. Dentro de QA de Desarrollo, se puede establecer un segundo nivel de QA, que es el realizado por el Gerente de Proyecto al probar conjuntamente el trabajo de los miembros del Equipo de Proyecto.

4.5.2 QA Interno

Cuando el Gerente de Proyecto, como responsable máximo del desarrollo del mismo, decide que una determinada parte de la aplicación, la totalidad de ella, está lista, la somete a la revisión de QA Interno, órgano independiente de Desarrollo como ya se ha visto anteriormente, quien intentará, por todos los medios a su alcance, *hacer fallar la aplicación que está probando*.

Esta es una de las diferencias más importantes entre los dos primeros niveles de QA. Lógicamente, un programador probar su programa con el deseo de que no falle, puesto que de eso se trata su trabajo, mientras que, por el contrario, la misión de QA Independiente es, precisamente, intentar conseguir el máximo de fallos posibles en el programa, ya que cuantos mas fallos localice, mejor estar desempeñando su labor. Como se puede ver, son dos puntos de vista diametralmente opuestos.

4.5.3 QA Independiente

Una vez que una aplicación, parte de ella, ha pasado el control de QA Interno satisfactoriamente, es enviada a QA Independiente.

Allí, es posible que se decida repetir ciertas pruebas ya realizadas por QA Interno, aunque lo más probable es que QA Independiente desarrolle sus propias pruebas, con criterios diferentes de los de QA Interno.

Es muy importante, pues, establecer un Plan de Pruebas para cada una de las Aplicaciones, en el que se indicará qué es lo que se va a probar, con qué rigurosidad, con qué datos, en qué orden, etc., y cada equipo de QA deberá de elaborar el suyo propio.

4.5.4 QA de Migración

Si QA Independiente encuentra algún procedimiento no satisfactorio, devolver la Aplicación a QA Interno, y éste a su vez a QA de Desarrollo, donde se corregirán los fallos y se repetir el proceso.

Si, por el contrario, QA Independiente decide que todo está correcto, y cuando toda la Aplicación haya sido probada, la enviar a QA de Migración.

Este organismo será el encargado de poner la Aplicación en un estado de Producción, llevando a cabo las labores pertinentes, para, una vez instalada la nueva Aplicación, poder realizar las pruebas finales de aceptación y las pruebas en paralelo con el Sistema actual, si existe, para verificar resultados durante un período razonable de tiempo, tras el cual, si todo funciona correctamente, desmontar los procedimientos de "vuelta atrás" que tendría previstos en caso de fallo y pasar el control efectivo de la Aplicación al Equipo de Producción.

4.6 Tareas de QA

Las tareas que un equipo de Control de Calidad debe de realizar, varían dependiendo de diversos factores: el nivel de QA, la fase del proyecto que se revisa, la severidad de QA, etc., y así, cada equipo de QA deberá de desarrollar sus propios tests de acuerdo con el estado en que se encuentre el proyecto. No obstante, el control de calidad del Software se basa preferentemente en revisar la Calidad de diseño Software, la Codificación, las Pruebas, la Documentación y finalmente la Auditoría, siguiendo para ello las normas que indique la metodología implantada en la instalación, que puede ser cualquiera de las muchas existentes en el mercado, y que marcar la pauta a seguir en el desarrollo de las pruebas del software. [1]

Por todo ello, es prácticamente imposible determinar cuáles serán las tareas más adecuadas a cada instalación en un momento determinado y bajo unas circunstancias concretas. No obstante, en las siguientes páginas y a modo de ejemplo, se citan algunas de las funciones más importantes de los primeros niveles de QA en cualquier instalación, (QA Interno y QA Independiente), sin entrar en detalles con respecto a QA de Desarrollo, por ser el menos formal de los tres, dejando para un capítulo aparte dada su importancia al último nivel de QA, el Control de Calidad de Migración de Aplicaciones a un estado de producción.

4.6.1 Funciones de QA

QA deberá de realizar las siguientes funciones:

[1] En este tema, la metodología a la que se hace referencia es PDM80 (Prototype Development Methodology 80), aunque es perfectamente válida cualquier otra metodología de desarrollo.

1 Revisar Estándares y acuerdos para diseño, codificación, pruebas y documentación.

2 Preparar formas y procedimientos para cada tarea a revisar, que serán usadas para informar de las deficiencias que se encuentren.

3 Revisar el diseño y documentación relacionada con el proyecto para verificar que se lleva a cabo en los lugares específicos durante el curso del desarrollo del sistema.

4 Revisar los listados de código.

5 Revisar los planes de pruebas y procedimientos acordados.

6 Revisar cualquier otra documentación (como manuales técnicos, manual del usuario, etc.)

7 Participar en las auditorías formales de la configuración.

8 Participar en las auditorías informales de la función de dirección de la configuración.

9 Planificar las transferencias de las responsabilidades de QA al personal de Producción encargado de operar el sistema.

4.6.2 Soporte de QA durante el desarrollo

QA deberá ser responsable de la preparación de formatos y procedimientos que se vayan a usar para las revisiones en cada etapa de desarrollo.

Cuando existan irregularidades o deficiencias, tiene que asegurarse de que los problemas encontrados son corregidos.

4.6.3 Revisiones principales

Las principales revisiones formales que se deben llevar a cabo durante el ciclo de vida del software son las siguientes:

4.6.3.1 Revisiones de Diseño

QA deberá de realizar las revisiones siguientes:

1 Revisión del estudio de posibilidades y requerimientos del sistema.

2 Revisión del diseño preliminar.

3 Revisión del diseño del sistema.

4 Revisión del diseño detallado del sistema.

El papel en todo este proceso es asegurar los Estándares y acuerdos de diseño y documentación, identificando informes y corrigiendo deficiencias.

El material a ser revisado para clarificación y adherencias a las normas Estándares podría incluir, entre otros los siguientes:

1 Planes de desarrollo del Software.

2 Asignación de tareas y autorización de procedimientos.

3 Planes de la dirección de la configuración.

4 Especificaciones de requerimientos.

5 Diseño de documentos.

6 Interfases con las identificaciones de diseño con otros sistemas.

7 Planes de diseño de implantación.

8 Software Estándar.

9 Planes de pruebas y procedimientos.

10 Planes de aceptación de pruebas.

11 Manuales del usuario.

4.6.3.2 Revisiones de Código

Cada listado de un programa deberá ser revisado tanto en su esencia como en su estilo, antes de su entrada en la librería de programas, así como periódicamente, ante posibles violaciones, y cada vez que sea modificado.
El material a ser revisado incluye:

1. Uso de lenguajes de programas no autorizados.
2. Uso de código no estructurado donde no sea permitido.
3. Violaciones de acuerdos de nombres (Bases de datos, programas, símbolos, sentencias, etc.)
4. Alineación en columnas impropias.
5. Violaciones de acuerdos de blancos ceros.
6. Formato incorrecto de mensajes.
7. Iniciación incompleta impropia de zonas de memoria.
8. Uso de técnicas de entradas y salidas prohibidas.
9. Insuficientes sentencias de comentarios en programas fuente.

4.6.3.3 Revisiones de "Testing"

QA deberá revisar todos los planes de integración, incluyendo planes de pruebas, procedimientos e informes, para verificar que cumplen los Estándares de "testing" acordados y atestiguar la ejecución de todas las pruebas, verificando sus resultados.

QA deberá investigar las violaciones de los Estándares aprobados en la integración de programas y testing. Ejemplos de estas posibles violaciones pueden ser:

1. Uso de métodos de integración de abajo hacia arriba.
2. Integración avanzada de unidades probadas.
3. Omisión del "test de regresión" cuando un módulo se añade al sistema al programa.

QA deberá identificar todos los ejemplos no conformes con el test durante la ejecución, examinando todos los listados para asegurar que los resultados Están correctamente grabados.

Asimismo QA deberá verificar que todas las pruebas requeridas Están, de hecho, efectuadas, documentadas y conformes.

4.6.3.4 Revisiones de documentación

Toda la documentación relacionada con un proyecto debe ser leída y revisada por QA para compararla con los Estándares y acuerdos establecidos. Dicha documentación incluirá:

1 Especificaciones del sistema.

2 Plan de configuración del sistema.

3 Plan de control de calidad del sistema.

4 Plan de desarrollo del sistema.

5 Programas de especificaciones de rendimiento.

6 Especificaciones de diseño de programas.

7 Especificaciones de diseño de interfases.

8 Listados de programas.

9 Especificaciones y planes de test.

10 Informes y procedimientos de test.

11 Plan de mantenimiento de manuales.

12 Manuales de usuario.

13 Plan de descripción de documentos.

Además, se revisará cada documento cuanto esté disponible para las tres etapas por las que atraviesa:

1 Bosquejo: Primera versión mecanografiada.

2 Preliminar: Versión para ser impresa.

3 Final: Versión para ser entregada.

4.6.3.5 Auditorías de la Configuración

QA ejecuta dos funciones durante las auditorías de la configuración. Actúa como soporte del personal de la configuración y ejecuta sus funciones usuales de comprobación para establecer Estándares y acuerdos.

Generalmente, se contemplan dos auditorías formales de la configuración casi al final del desarrollo del sistema y antes del paso a producción, una *Auditoría de la Configuración Funcional*, en la que el usuario suministra una lista de ítems a ser auditada, y una *Auditoría de la Configuración Física*, donde se confirma que los ítems están disponibles en cada componente para ser utilizados.

4.6.3.6 QA durante la etapa de Operación / Mantenimiento

Aunque la participación de QA se reduce durante el período de Operación/Mantenimiento de la Aplicación, sin embargo debe revisar las actualizaciones del producto, siendo responsabilidad de QA verificar que todas las modificaciones de diseño son implementadas y documentadas según los Estándares aplicados, así como las modificaciones añadidos al código.

4.7 Control de Calidad de Migración

Es, sin duda, el nivel más importante de Control de Calidad en cualquier instalación, y del que no se debería de prescindir en ninguna circunstancia. Recibe diversos nombres dependiendo de la organización que lo implante, pudiendo conocerse como Equipo de Aceptación de Aplicaciones, Validación de Aplicaciones, etc., pero sus funciones son prácticamente las mismas y deben de cubrir unos mínimos que garanticen el correcto funcionamiento del software antes de pasar a un estado de producción con trabajo real. Es, en definitiva, quién deberá de dar las últimas bendiciones al software antes de que empiece a "rodar" definitivamente.

A continuación se detallan de forma bastante exhaustiva los detalles sobre quienes lo forman, qué tareas deben de realizar y en qué momento, qué requisitos debe de cumplir el software para que sea aceptado para su revisión por QA de Migración, etc., etc.

4.7.1 Equipo de QA de Migración

El Equipo de Control de Calidad de Migración de Aplicaciones a Producción (MQA: Migration Quality Assurance) estará formado por personal del Departamento del Usuario, de Explotación (Producción), de QA Independiente, y de Administración de Bases de Datos. La responsabilidad de coordinación de las actividades descritas en todos los procedimientos que a continuación se detallan, recae en el responsable de Control de Calidad Independiente.

Dicho responsable, que a partir de ahora se conocerá como Coordinador de MQA, será el encargado de establecer los contactos necesarios con los departamentos antes mencionados para determinar qué personas quedan adscritas al Equipo de MQA, para ser utilizadas en el momento en que su participación sea necesaria.

4.7.2 Elaboración del Plan de Pruebas

Consideraremos aquí dos tipos de pruebas diferentes que deben de ser planificadas por separado: las pruebas realizadas por el Usuario Final, para dar conformidad a la Aplicación y las Pruebas de Entorno, para comprobar determinados aspectos de la Aplicación que pueden no haber sido probados anteriormente.

4.7.2.1 Pruebas del Usuario

Una vez terminada la fase de Desarrollo, el Coordinador de MQA, en colaboración con el Usuario, comienza a preparar un Plan de Pruebas de Migración, el cual variar dependiendo del Proyecto de que se trate, pero que, en general, deberá de crear situaciones propias e independientes, utilizando como base la documentación de Requerimientos del Usuario.

Por otra parte, servir n como guía para la confección de los tests respectivos el Check-List del Plan de Pruebas, en su totalidad, y el Check-List de Programación, este último solo para verificar condiciones extremas de la Aplicación. (Ver Modelos MQA01 y QA12 en el Anexo)

Este Plan deberá de incluir:

1 Tests para comprobación de la calidad y fiabilidad de datos y resultados

2 Estimación de tiempos de Ordenador necesarios

3 Determinación de segmentos críticos

4 Pruebas de la Aplicación, incluyendo:

Prueba Individual de Transacciones:

a) Con condiciones válidas
b) Con condiciones inválidas
c) Con contingencias (mezcla de las anteriores)

Pruebas de Integración del Batch

a) De funcionamiento
b) De resultados

Pruebas de Ciclos

a) Periódicos: Diario, semanal, mensual, anual, etc.
b) No periódicos

4.7.2.2 Pruebas de Entorno

El Coordinador de MQA y Programación de Sistemas elaboran tests para verificar los siguientes puntos:

1 Rendimiento del Sistema

2 Posible impacto en otras Aplicaciones

3 Seguridad del Sistema

4 Capacidades de Backup

5 Posibilidad de Recovery

4.7.3 Requisitos para la Migración

Antes de que una Aplicación se considere lista para comenzar el proceso de Migración a Producción, el Coordinador de MQA deberá de verificar que se cumplen los siguientes requisitos:

1 Que todos los elementos de hardware necesarios Están disponibles

2 Que la Aplicación cumple los requerimientos aprobados por el Usuario y las especificaciones de Diseño del Sistema

3 Que Control de Calidad Independiente ha revisado y aprobado (ver Mod. QA10) los Estándares de:

Esquemas
Subesquemas
Programas fuente
Pantallas
Transacciones
Librerías
Explotación
Documentación
Diccionario de Datos
Metodología

4 Que Control de Calidad Independiente ha comprobado los resultados de las pruebas de Desarrollo de:

Programas de actualización de la Base de Datos
Programas de creación, carga y descarga de la Base de Datos
Programas de verificación de la Base de Datos
Simulaciones de pruebas on-line de todos los componentes
Programas de reorganización de la Base de Datos
Integración de módulos

5 Que Control de Calidad Independiente ha realizado:

Pruebas Individuales:

a) De programas en tiempo real
b) De programas "Batch"

Pruebas de Transacciones

Pruebas de Integración del Batch
Pruebas de Bases de Datos:

a) Carga
b) Descarga
c) Reorganización
d) Integridad

6 Que la Aplicación tiene incorporados todos los requisitos de:

Seguridad del Sistema:

a) Existencia de Plan de Backup y Recovery

Integridad de Datos:

a) Existencia de cuenta de artículos en actualizaciones batch

b) Existencia de cuadres contables en actualizaciones batch y teleproceso, si procede

c) Existencia de tests para verificación de programas de entrada y actualización de datos

Nivel de Servicio:

a) Acuerdos Usuario-Producción

Documentación:

a) Del Usuario
b) De Producción

Formación:

a) Del Usuario
b) De Producción

Volúmenes de Datos
Copias de Seguridad
Programas de verificación de la carga de la Base de Datos

7 Que se ha especificado la fecha tentativa de implementación

4.7.4 *Definición de Fechas y Actividades*

Una vez verificado que se cumplen los requisitos anteriores, el Coordinador de MQA convoca una reunión en la que participan:

El Equipo de MQA
Coordinación de Aplicaciones
Control de Cambios
Programación de Sistemas
El Gerente del Proyecto

En esta reunión se estudian los aspectos que figuran en los puntos siguientes.

4.7.4.1 Fecha Tentativa de Implementación y Plan de Pruebas

Se estudia la fecha tentativa de implementación, para determinar si es factible realizar todas las pruebas del Plan dentro de dicha fecha.
De no ser así, se vería la posibilidad de retrasar la fecha de implementación, y en el caso de no ser posible dicho retraso, se supeditarían la rigurosidad y cantidad de las pruebas a la fecha establecida.

4.7.4.2 Ordenador de Desarrollo y/o Ordenador de Producción

Se decide si procede realizar las primeras pruebas en el ordenador de Desarrollo, si, por el contrario, éstas se llevan a cabo en el ordenador de Producción.

4.7.4.3 Calendario de Pruebas

Se confecciona un calendario de pruebas basándose en el Plan de Pruebas y a la Fecha de Implementación prevista. (Modelo MQA05)

4.7.4.4 Matriz Funciones-Personas

Se elabora una matriz, indicando cada una de las funciones a realizar y asignando las personas correspondientes. (Modelo MQA05)

Los resultados de esta reunión quedar n reflejados en un acta con la conformidad reparos de los asistentes, la cual se hará seguir a la Dirección.

4.7.5 Test de Aceptación

A partir de este momento, el Equipo de MQA comienza a realizar las pruebas estipuladas en el Plan, para lo cual lleva a cabo los siguientes pasos:

a) Preparación:

Aporta los parámetros necesarios para solicitar de Programación de Sistemas la generación de un Sistema (SYSGEN) de Producción, si fuera necesario.

Hace un backup de todo el entorno del sistema a implementar.

Obtiene las Librerías de Transición de MQA a partir de las Librerías de QA Independiente, vía compilación, utilizando el Check-List de Migración. (Modelo MQA06)

Revisa los JCL (Job Control Language) de Producción.

Verifica y carga los datos necesarios en sus Bases de Datos de Transición.

Verifica los segmentos críticos.

b) Ejecución:

Ejecuta las pruebas del Plan.

Simultáneamente con la ejecución de las pruebas, el Equipo de MQA verifica el Check-List de Aceptación de Aplicaciones. Este Check-list deberá de comprobar que:

El Sistema responde a los requisitos funcionales definidos por el Usuario.

El Sistema rinde dentro de los límites aceptables de:

- Respuesta (Individualmente e integrado)
- Trabajo producido en relación con Tiempo de ejecución
- Consumo de recursos

El Equipo de Producción es capaz de operar el sistema solamente con el Manual de Producción

Son adecuadas y funcionales las características de:

- Seguridad
- Integridad
- Prevención del fraude
- Auditabilidad

Se han incluido todos los Programas de Utilidad necesarios para:

- Carga y descarga de la Base de Datos
- Copias de Seguridad (Backup)

- Reorganización
- Recuperación (Recovery)
- Relanzamiento
- Mantenimiento

Los programas anteriores funcionan adecuadamente y Están correctamente documentados

Hay previstos procedimientos para situaciones fuera de calendario y emergencia

Se cumplen los Estándares de Explotación relativos a:

- Picos y promedios de utilización
- Grado de impacto con el resto de sistemas on-line
- Copias de seguridad del Sistema
- Datos y volúmenes de datos
- Documentos de salida
- Mensajes sobre Consola
- Soportes de Entrada
- Fechas y captura de datos externos
- Ordenes de Trabajo
- Códigos de Abort del Usuario
- Documentación de dependencia de Jobs
- Estimación de tiempos por Procesos
- Estimación de líneas de impresión
- Verificación de Nivel de Servicio aprobado y firmado

La eliminación lógica y física de artículos es correcta

Es correcta la baja de Sets de Base de Datos con miembros dependientes de dichos Sets

La distribución física de los datos es equilibrada

De acuerdo con los resultados obtenidos en las pruebas, y de conformidad con el Usuario, el Coordinador de MQA decidirá, bien devolver la Aplicación a QA Independiente para su revisión bien dar el visto bueno a la misma.

4.7.6 Fecha de Instalación Real

Una vez superados los Test de Aceptación, el Coordinador de MQA convocar una reunión en la que participará:

Gerente de Proyecto
Explotación
Control de Cambios
Coordinación de Aplicaciones
Administración de Bases de Datos
El Departamento del Usuario

La finalidad de esta reunión es notificar el final de los Tests de Aceptación y determinar la fecha idónea de Puesta en Marcha Real de la Aplicación para someterla a la aprobación de la Dirección.

4.7.7 Puesta en Producción Definitiva

Una vez determinada la fecha de instalación real de la Aplicación, Control de Cambios notifica con la antelación suficiente a Explotación (Producción) que puede realizar el pase definitivo de todos los componentes de la Aplicación a las Librerías de Producción a partir de las Librerías de Transición de MQA (vía copia, utilizando el Check-List de Migración)

Explotación realiza un backup de las Bases de Datos, Librerías y Ficheros de Transición de MQA, así como del Juego de Implementación. A continuación, lleva a cabo la carga inicial real de las Bases de Datos y de los ficheros de Producción, si procede. Completado el Check-List de Migración, Explotación lo devuelve a Control de Cambios para su revisión y archivo.

Después de haberse integrado el nuevo Sistema en el entorno existente, se deja rodar un día sin realizar ningún tipo de trabajo con el nuevo Sistema, con el fin de determinar posibles problemas potenciales de integración.

Seguidamente, se llevan a cabo pruebas de trabajo real, las cuales variar n dependiendo del tipo de Aplicación de que se trate, ejecutándose pruebas en paralelo mientras sea aconsejable.

4.7.8 Revisión de Post-implementación

Dentro de la semana siguiente a la puesta en marcha real de la Aplicación, se realiza una revisión de post-implementación por parte de MQA, Administración de Datos y Programación de Sistemas con el fin de:

1 Documentar cualquier problema o solución no descubierta durante la fase de operación

2 Evaluar los procedimientos de Pruebas-Producción

Tras un período razonable de operaciones sin problemas, Programación de Sistemas archiva los componentes de "marcha atrás" especificados en el test de regresión y MQA retira el Juego de Implementación de sus Librerías de Transición.

Con esto finalizan, por el momento, las labores de MQA, que, como se puede apreciar, son determinantes para el buen funcionamiento de la Aplicación, y para conseguir la satisfacción de los usuarios, que es en definitiva lo que cualquier organización debe de perseguir.

4.8 Control de Versiones del Software

Otro de los mecanismos que se deben de implantar en un entorno de QA es el de Control de Versiones del Software. Este mecanismo mostrará el camino que deberá de recorrer el software antes de su paso a Producción, indicando los niveles de control de calidad que deberá de atravesar, y determinando también cuales son los métodos de vuelta atrás en el caso de encontrarse fallos.

Existen dos etapas perfectamente marcadas en el Ciclo de Vida del Software: La Etapa de Desarrollo y la Etapa de Producción. El control del software en estas dos etapas es diferente, ya que la severidad a aplicar sobre las modificaciones al software una vez en producción tiene que ser infinitamente mayor que la aplicada sobre el software mientras éste se está desarrollando.

En algunos casos, no se realiza control de versiones mientras el software está en desarrollo. No obstante, cada organización deberá de definir sus procedimientos de modificación en ambos casos, extremando las precauciones cuando se trate de software que esté funcionando en trabajo real. A continuación se muestra una posible implementación de un Control de Versiones.

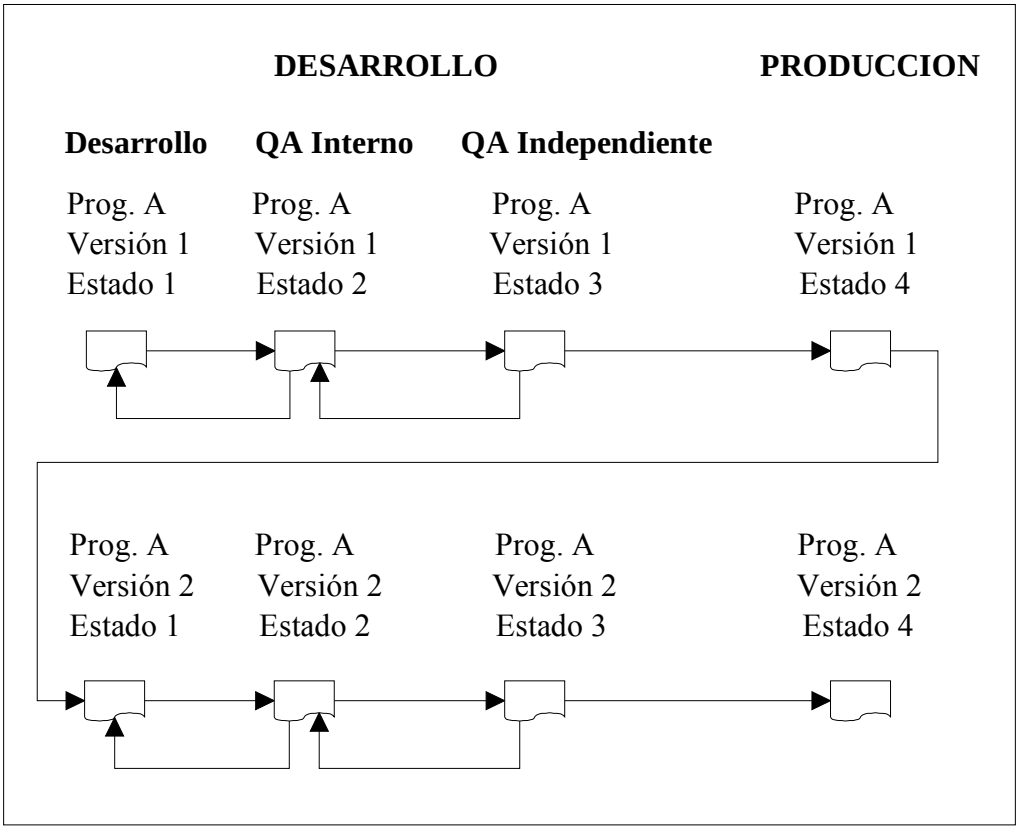

Fig. 4.2 Control de Versiones del Software

4.9 Otros aspectos de Control de Calidad

Hasta aquí, hemos pasado revista a una serie de mecanismos que nos permiten efectuar un control sobre el software desarrollado en un Centro de Proceso de Datos en una empresa de servicios, de cualquier otra que desee desarrollar sus propias aplicaciones, y que serán mas menos efectivos dependiendo del grado de severidad con el que se quieran aplicar. Pero el concepto de QA es todavía mas amplio y no se limita, como pudiera desprenderse por lo hasta ahora mencionado, al control del desarrollo del software, sino que debe de abarcar la totalidad de los procesos de la Instalación.

Estamos tratando de crear un *entorno de calidad*, y eso no se consigue solamente controlando a los programadores. Habrá que crear el ambiente necesario, establecer las motivaciones oportunas y prever los mecanismos de corrección para eliminar las desviaciones cuando estas se presenten. Es mas, el control de calidad deberá de complementarse con mecanismos tales como un efectivo *Control de Cambios*, que afectar fundamentalmente al hardware, un correcto *Control de Problemas*, donde se resolverán las quejas cuando aparezcan y se llevará control de las incidencias para poder predecir posibles fallos en el futuro, un *Control de Recursos*, que nos indicar los recursos disponibles en cada momento, etc., etc.

La idea que subyace detrás de todo esto es la mentalización de todas las personas hacia la realización de un trabajo de calidad; es preciso repetir una vez mas que hay que tender a *cero errores*, aunque de antemano se sepa que esto no es posible, pero la intención sí debe de estar en la mente de todas las personas cuando se acomete cualquier trabajo.

Y esto no se conseguir sin el esfuerzo conjunto de todos (Dirección incluida) y sin la aceptación voluntaria y con entusiasmo de todas las técnicas hasta ahora comentadas.

4.10 Consideraciones finales

Para finalizar este capitulo sobre Control de Calidad de Desarrollo Sistemas, solo queda por añadir unos breves comentarios que puedan servir de ayuda para conseguir el entorno de calidad antes mencionado y que entendemos es el que se debe de buscar.

En primer lugar, deberemos de tener en cuenta que ninguna opción de QA se podrá implantar con éxito *sin el total apoyo de la Alta Dirección de la Empresa*. La carga de trabajo y el costo que la implantación supone hacen inviable el intento si el proceso no es "top-down".

Por lo tanto, será necesario comenzar por "venderle la idea" a la dirección, para garantizar que seremos capaces de contar con el apoyo necesario para la implantación correcta del entorno que deseamos.

En segundo lugar, una última consideración: este capítulo no pretende (ni puede) cubrir todos los mecanismos de QA, la forma de su implantación, los "check-lists" que utiliza, los planes de prueba que debe de desarrollar, etc., los cuales requerirían para su explicación una extensión bastante superior a la aquí dedicada. No obstante, existen en el mercado una gran cantidad de libros especializados en

el tema, que podrán ser de gran utilidad a las personas que decidan emprender en sus respectivas organizaciones el difícil camino de la calidad.

4.11 Anexos

En las páginas siguientes figuran una serie de Modelos de Impresos de uso común en cualquier instalación que sirven de apoyo al desarrollo de las labores de Control de Calidad que hemos revisado.

Se muestran aquí a modo de ejemplo, aunque lo mas aconsejable, y lo más probable, es que cada organización desarrolle los suyos propios.

MQA MIGRACION DE APLICACIONES A PRODUCCION

Check-List

Proyecto: Fecha:
Gerente de Proyecto:
Preparado por: Hoja 1 de 1

FUNCION	Fecha Comienzo	Fecha Final	Observaciones
1.- Equipo de Control de Calidad de Migración 2.- Elaboración del Plan de Pruebas 2.1 Pruebas del Usuario 2.2 Pruebas de Entorno 3.- Requisitos para la Migración 4.- Definición de Fechas y Actividades 4.1 Fecha Tentativa de Implementación y Plan de Pruebas 4.2 Ordenador de Desarrollo y/o Ordenador de Producción 4.3 Calendario de Pruebas 4.4 Matriz Funciones - Personas 5.- Test de Aceptación 6.- Determinación de la Fecha de Instalación 7.- Puesta en Producción definitiva 8.- Revisión de Post-implementación			

Mod. MQA00

MQA PLAN DE PRUEBAS

Check-List

Proyecto: Fecha:
Gerente de Proyecto:
Preparado por: Hoja 1 de 2

PUNTO A VERIFICAR	OK	OBSERVACIONES
COMPROBAR EXISTENCIA DE:		
Tests de comprobación de: 1.- Calidad de Datos y Resultados 2.- Fiabilidad de Datos y Resultados Prueba individual de Transacciones: 1.- Con condiciones válidas 2.- Con condiciones inválidas 3.- Contingencias Pruebas de Integración del Batch: 1.- De funcionamiento 2.- De resultados Pruebas de Ciclos Periódicos: 1. Diario 2. Semanal 3. Quincenal 4. Mensual 5. Trimestral 6. Semestral 7. Anual 8. Otro período Pruebas de Ciclos No Periódicos Pruebas de Entorno: 1.- De Rendimiento 2.- De Impacto en el Sistema antiguo 3.- De Seguridad 4.- De Backup 5.- De Recovery Estimación de tiempos de Ordenador Determinación de segmentos críticos		

Mod. MQA01

MQA PLAN DE PRUEBAS

Check-List

Proyecto: Fecha:
Gerente de Proyecto:
Preparado por: Hoja 2 de 2

PUNTO A VERIFICAR	OK	OBSERVACIONES
CONDICIONES A COMPROBAR EN LOS TESTS		
Variaciones: 1.- Altas con correspondencia 2.- Bajas sin correspondencia 3.- Modificaciones sin correspondencia 4.- Altas, bajas y modificaciones repetidas Campos numéricos: 1.- Cambio de valor de + a - y viceversa 2.- Movimientos de datos de campos alfanuméricos sobre campos numéricos 3.- Forzar datos para verificar truncamientos y condiciones de overflow Tablas: 1.- Forzar su capacidad para verificar posibles situaciones de abort Ficheros e Impresos: 1.- Tratamiento del primer y último registro 2.- Falta de cabeceras, colas y totales en las cintas 3.- Procedimientos multibobina y multifichero 4.- Continuidad de la numeración de páginas en los informes Entrada de Datos: 1.- En general, forzar todos los datos de entrada para verificación 2.- Verificar formatos de fechas		

Mod. MQA01

MQA PLAN DE PRUEBAS

Proyecto: Fecha:
Gerente de Proyecto:
Preparado por:
Módulo (JCL, Programa, Transacción): Hoja 1 de 1

Funciones	Tiempos de Ejecución		Resultados	Observaciones
	Previsto	Real		

Mod. MQA02

MQA REQUISITOS

Check-List

Proyecto: Fecha:
Gerente de Proyecto:
Preparado por: Hoja 1 de 3

PUNTO A VERIFICAR	OK	OBSERVACIONES
GENERAL: Disponibilidad del hardware Cumplimiento de Requerimientos aprobados Cumplimiento Especificaciones de Diseño Existencia Fecha Tentativa de Implementación **REVISION Y APROBACION DE IQA DE:** Estándares de: 1.- Esquemas 2.- Subesquemas 3.- Programas fuente 4.- Pantallas 5.- Transacciones 6.- Metodología PDM80 7.- Librerías 8.- Explotación 9.- Documentación 10.- Diccionario de Datos Resultados de las pruebas de Desarrollo de programas sobre la Base de Datos, de: 1.- Creación, carga y descarga 2.- Actualización 3.- Verificación 4.- Reorganización 5.- Simulaciones On-line 6.- Integración de Módulos		

Mod. MQA03

MQA REQUISITOS

Check-List

Proyecto: Fecha:
Gerente de Proyecto:
Preparado por: Hoja 2 de 3

PUNTO A VERIFICAR	OK	OBSERVACIONES
REALIZACION DE IQA DE: Pruebas individuales de: 1.- TPR's 2.- Batch Pruebas de Transacciones Pruebas de Integración del Batch Pruebas sobre Base de Datos de: 1.- Carga 2.- Descarga 3.- Reorganización 4.- Integridad **INCORPORACION EN LA APLICACION DE LOS REQUISITOS DE:** Seguridad del Sistema: 1.- Existencia de Plan de Backup y Recovery Integridad de Datos: Deberá existir: 1.- Cuenta de artículos en Actualizaciones Batch 2.- Cuadres contables, si procede, en Actualizaciones Batch y T.P. 3.- Tests de verificación de programas de entrada y actualización de datos		

Mod. MQA03

MQA REQUISITOS

Check-List

Proyecto: Fecha:
Gerente de Proyecto:
Preparado por: Hoja 3 de 3

PUNTO A VERIFICAR	OK	OBSERVACIONES
INCORPORACION EN LA APLICACION DE LOS REQUISITOS DE: Nivel de Servicio: 1.- Acuerdos Usuario-Producción Documentación: 1.- Del Usuario 2.- De Producción Formación: 1.- Del Usuario 2.- De Producción Volúmenes de Datos Copias de Seguridad Programas de Verificación de carga y descarga de la Base de Datos		

Mod. MQA03

MQA TEST DE ACEPTACION

Check-List

Proyecto: Fecha:
Gerente de Proyecto:
Preparado por: Hoja 1 de 2

PUNTO A VERIFICAR	OK	OBSERVACIONES
PREPARACION DEL PLAN DE PRUEBAS		
Solicitar SYSGEN de Producción		
Hacer Backup de todo el entorno		
Obtener Librerías de Transición MQA a partir de IQA (Vía compilación)		
Revisar JCL de Producción		
Verificar y cargar B. Datos de Transición		
Verificar los segmentos críticos		
EJECUCION DEL PLAN DE PRUEBAS. VERIFICAR:		
Procedimientos de emergencia		
Procedimientos fuera de calendario		
Estándares de Explotación: 1.- Documentos de Salida 2.- Mensajes sobre Consola 3.- Soportes de Entrada 4.- Fechas 5.- Captura de datos externos 6.- Ordenes de Trabajo 7.- Códigos de Abort del Usuario 8.- Documentación dependencia de Jobs 9.- Estimación tiempos por procesos 10.- Estimación de líneas de impresión 11.- Verificación de Nivel de Servicio 12.- Comprobación de versiones de ficheros según disposiciones legales		

Mod. MQA04

MQA

TEST DE ACEPTACION

Check-List

Proyecto: Fecha:
Gerente de Proyecto:
Preparado por: Hoja 2 de 2

PUNTO A VERIFICAR	OK	OBSERVACIONES
EJECUCION DEL PLAN DE PRUEBAS. VERIFICAR: Copias de Seguridad Datos y volúmenes de datos Documentos de salida: 1.- Resultados correctos 2.- De acuerdo con los Estándares Eliminación lógica y física de artículos Bajas de Sets con miembros dependientes Distribución física de los datos **COMPROBAR QUE:** El Sistema: 1.- Responde a los Requerimientos 2.- Rinde aceptablemente en: - Tiempo respuesta (individual) - Tiempo respuesta (integrado) - Consumo de recursos - Picos y promedios de uso - Grado impacto con resto del TP. - Trabajo Producido/Tiempo Ejecución 3.- Incorpora las características de: - Seguridad - Integridad - Prevención del fraude - Auditabilidad 4.- Incluye programas de Utilidad de: -Carga/descarga de Bases de Datos - Backup - Reorganización - Recuperación - Relanzamiento - Mantenimiento		

MQA MATRIZ FUNCIONES - PERSONAS

Proyecto: Fecha:
Gerente de Proyecto: Hoja 1 de 1

Funciones	Dpto.	Persona Asignada	Fecha Prevista Comienzo	Tiempo Estimado Ejecución	Observaciones

Mod. MQA05

Visto Bueno:

Programación de Sistemas Coordinación de Aplicaciones

Gerente Proyecto Control Cambios MQA

MQA MIGRACION DEL SISTEMA

Check-List

Proyecto: Fecha:
Gerente de Proyecto:
Preparado por: Hoja 1 de 1

De IQA a MQA ☐ De MQA a Producción ☐

FUNCIONES	FECHA	OK
Backup de: 1.- Librerías - Programas - Jobs - Rutinas - Utilidades 2.- Ficheros 3.- Base de Datos - Esquemas - Subesquemas 4.- Juego de Ensayo Reserva de: 1.- Base de Datos 2.- Ficheros 3.- Jobs 4.- Librerías 5.- Utilidades Carga de: 1.- Base de Datos 2.- Ficheros 3.- Jobs 4.- Librerías 5.- Utilidades Varios:		
Firmado:		

Mod. MQA06

CONTROL DE CALIDAD DE ESTANDARES DE PROGRAMACION

PC: Primera compilación
RF: Revisión Final

Pasa Q.A. Si ☐ No ☐

Nombre del Programa: Fecha PC:
Programador: Fecha RF.
Revisado por: Hoja 1 de 5

PUNTO A VERIFICAR	PC			RF		OBSERVACIONES
	S	N		S	N	
COMPROBAR EXISTENCIA DE:						
Especificaciones completas						
Copias de Subrutinas						
SADT y VTOC						
J.C.L. (Job Control Language)						
Plan de Test						
Datos del Test						
Codificación del Programa						
Prueba del Programa						
DESCRIPCION DE FICHEROS:						
Copiados de Librería ó Diccionario de Datos						
PIC y VALUE en Col. 32 y 44						
Consistencia de Nombres						
Números de Nivel adecuados						
Alineación de Números de Nivel						
Separación entre FD's con *						

Mod. QA12

CONTROL DE CALIDAD DE ESTANDARES DE PROGRAMACION

PC: Primera compilación
RF: Revisión Final

Pasa Q.A. Si ☐ No ☐

Nombre del Programa: Fecha PC:
Programador: Fecha RF.
Revisado por: Hoja 2 de 5

PUNTO A VERIFICAR	PC			RF		OBSERVACIONES
	S	N		S	N	
WORKING-STORAGE SECTION:						
Sufijo -W (Excepto Switches)						
Consistencia de Nombres						
Alineación de Niveles						
Agrupamiento de campos						
PIC y VALUE en Col. 32 y 44						
Separación con *						
Uso de Niveles 88						
Uso de Switches (Sufijo -SW)						
REPORT SECTION:						
Alineación						
PIC y VALUE en Col. 32 y 44						
Separación con *						
Números de Nivel no correlativos						
Referencia a Programa y Job						

Mod. QA12

CONTROL DE CALIDAD DE ESTANDARES DE PROGRAMACION

PC: Primera compilación
RF: Revisión Final

Pasa Q.A. Si ☐ No ☐

Nombre del Programa: Fecha PC:
Programador: Fecha RF.
Revisado por: Hoja 3 de 5

PUNTO A VERIFICAR	PC			RF		OBSERVACIONES
	S	N		S	N	
PROCEDURE DIVISION: General						
Estructura correcta						
Nombre de Módulos						
Numeración de Módulos						
Documentación de Módulos						
Separación de Módulos con *						
Longitud de Módulos (Máx. 50L.)						
Independencia de Módulos						
PROCEDURE DIVISION: Alineación						
Open y Close						
Read y Write						
Nidos IF						
Condiciones compuestas						
Instrucciones muy largas						
Instrucciones Imperativas						

Mod. QA12

CONTROL DE CALIDAD DE ESTANDARES DE PROGRAMACION

Revisión Final

Pasa Q.A. Si ☐ No☐

Nombre del Programa: Fecha:
Programador: Revisión:
Revisado por: Hoja 4 de 5

PRUEBAS DEL PROGRAMA	OK		OBSERVACIONES
	S	N	

Visto Bueno: **Fecha:**

Mod. QA12

CONTROL DE CALIDAD DE ESTANDARES DE PROGRAMACION

Nombre del Programa: Fecha:
Programador: Revisión:
Revisado por: Hoja 5 de 5

OBSERVACIONES

Mod. QA12

CONTROL DE CALIDAD de ESTANDARES DE PROGRAMACION

C H E C K - L I S T de P R G R A M A S (Orientado a COBOL)

Errores de referencia de Datos:

1.- ¿Hay variables sin inicializar?

2.- ¿Hay algún índice fuera de rango?

3.- ¿Hay algún índice con valor cero?

4.- ¿Hay algún índice con valor no entero?

5.- Al usar REDEFINES, ¿se corresponden las PICTURES con los datos?

6.- ¿Se corresponden los datos de los ficheros con su descripción?

7.- ¿Se han verificado todos los truncamientos?

8.- Si se usa una estructura de datos en varios procesos, ¿es esta estructura idéntica en todos ellos?

9.- ¿Tienen las tablas la capacidad suficiente?

10.- En operaciones con tablas, ¿se procesa correctamente el último dato?

Errores de Declaración de Datos:

1.- ¿Tienen todos los campos los nombres adecuados?

2.- ¿Se han asignado PICTURES correctas a las variables utilizadas? Por ejemplo, ¿se utilizan campos numéricos sólo si se van a efectuar operaciones aritméticas con ellos?

Errores de Operaciones:

1.- ¿Se han revisado las posibles condiciones de overflow y underflow?

2.- Cuando sea aplicable, ¿puede el valor de una variable salirse de su rango significativo? ¿Están debidamente controlados los rangos de las variables?

3.- ¿Se usa correctamente la jerarquía de operaciones aritméticas? ¿Se utilizan correctamente los paréntesis?

4.- ¿Se usa correctamente la jerarquía de operaciones booleanas?

Errores de Comparación:

1.- ¿Hay alguna comparación entre variables de distinta clase de distinta longitud?

2.- ¿Se utilizan correctamente los operadores de comparación?

Errores de Flujo de Datos:

1.- ¿Hay siempre una condición de salida de los bucles?

2.- ¿Hay una condición para el final del programa?

3.- ¿Es posible que debido a las condiciones de un bucle este nunca se ejecute? Si es así, ¿es esto correcto?

4.- ¿Tiene cada IF su correspondiente ELSE?

5.- ¿Hay alguna decisión no exhaustiva? Por ejemplo, si un campo puede valer 1, 2 ó 3, ¿asume la lógica que sea 3 si no es ni 1 ni 2? Si es así, ¿es esta asunción válida?

Errores de Interfase:

1.- ¿Es correcto el orden de los argumentos de entrada y salida de un programa llamado con CALL?

2.- ¿Se interpretan correctamente las unidades en que Están expresados los argumentos? Por ejemplo, ¿nos devuelve una rutina una fecha en días (fecha Juliana) y en el programa se interpreta como DDMMAA?

3.- ¿Es el número de argumentos pasados a un módulo el que este necesita? ¿Se pasan todos los argumentos al módulo?

4.- Si un módulo tiene distintos puntos de entrada, ¿está el programa entrando en el punto correcto?

5.- Si hay variables compartidas por mas de un módulo, ¿tienen la misma definición en cada uno de ellos?

6.- ¿Puede una subrutina alterar el valor de un campo que no deba de ser alterado?

Errores de I/O:

1.- ¿Están los atributos de los ficheros correctamente declarados? ¿Los posibles valores asumidos por defecto son los deseados?

2.- ¿Son los atributos del OPEN correctos? ¿Están todos los ficheros abiertos antes de ser usados?

3.- ¿Se detectan y se manipulan correctamente las condiciones de fin de fichero?

4.- ¿ Están los textos de los DISPLAYS de los impresos escritos correctamente?

Otros puntos de chequeo:

1.- ¿Se utilizan todas las variables referenciadas en la lista de referencias cruzadas? Si no es así, ¿es correcto que se hayan definido variables que no se utilizan?

2.- ¿Se han revisado todos los mensajes del Compilador, inclusive los de WARNING?

4.12 Indice de Abreviaturas y Glosario de Términos

Se dan a continuación el glosario de términos y las abreviaturas empleadas en este tema.

JCL	Job Control Language (Lenguaje de Control de Jobs)
Job	Tarea, conjunto de programas de una Aplicación
PDM80	Prototype Development Methodology 80 (Metodología de Desarrollo de Prototipos 80)
T.P.	Teleproceso
SADT	System Analysis and Design Techniques (Técnicas de Diseño y Análisis de Sistemas)
SYSGEN	System Generation (Generación del Sistema)
VTOC	Visual Table of Contents (Tabla Visual de Contenidos, utilizada para mostrar los diagramas de bloques de un programa)

4.13 Bibliografía

[Bentley, 1982] Colin Bentley. *Computer Project Management*. C. Heyden & Son Ltd. 1982.

[Boehm, 1981] Barry W. Boehm. *Software Eingineering Economics*. Prentice Hall. 1981.

[Cárdenas, 1985] Alfonso Cárdenas. *PDM80. Prototype Development Methodology*. Computomata Inlt. Co. 1985.

[Caridad, 1991] Serafín Caridad. *La Importancia de Control de Calidad en la Gestión de Proyectos Inform ticos*. Separata de El Reto de la Inform tica en la Década de los Noventa. Publicaciones de la Fundación Alfredo Brañas. Velograf. S.A. 1991.

[Caridad-Souto, 1985] Serafín Caridad y Ramón Souto. *Control de Calidad de Seguimiento de Proyectos.* Banco Pastor, S. A. 1985.

[Connell-Brice, 1985] John Connel y Linda Brice. *Practical Quality Assurance*. Datamation. Marzo, 1985.

[DeMarco, 1982] Thomas DeMarco. *Controlling Software Projects*. 1982

[Grafton, 1986] William Grafton. *Test to Production Migration Procedures For DB/DC*. Computomata Intl. Co. 1986.

[Peat, 1980] Peat, Marwick, Mitchell & Co. *System Development Manual.* 1980.

[Souto, 1986] Ramón Souto. *Procedimiento de Migración de Aplicaciones a Producción*. Banco Pastor, S. A. 1986.

4.14 Prácticas

Efectuar el Control de Calidad de cualquier Aplicación de la que se disponga.