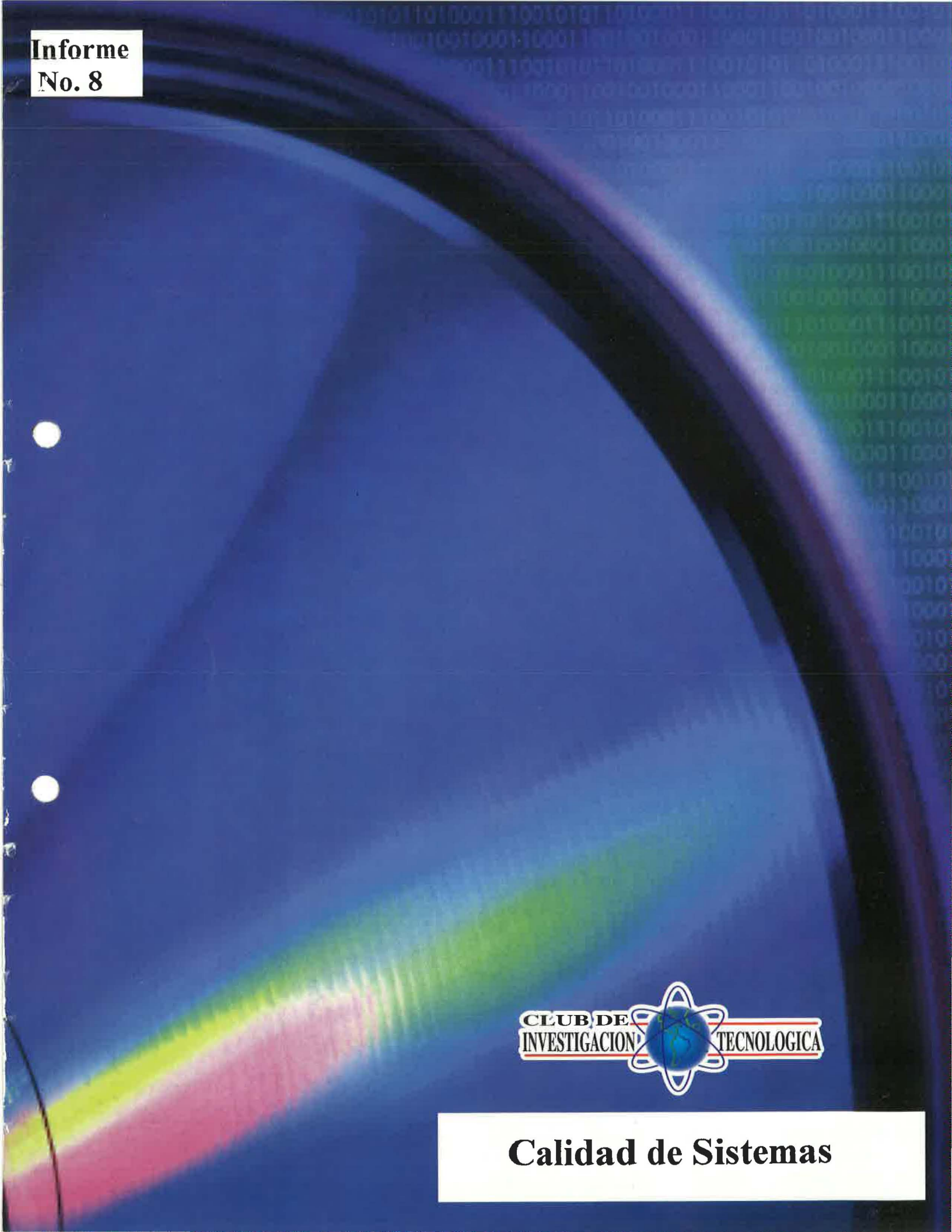
Informe
No. 8



Calidad de Sistemas

Club de Investigación Tecnológica

Calidad de Sistemas

Preparado por: Ulises Agüero Arroyo
Octubre 1990

Editado y Publicado por Rho-Sigma, S.A.
a nombre del Club de Investigación Tecnológica.
Todos los derechos reservados.
Prohibida la reproducción total o parcial.
San José, Costa Rica
Mayo 1990

Resumen Ejecutivo

La calidad de los sistemas de software se torna cada día más importante debido a que las inversiones en sistemas continuan creciendo tanto en valor absoluto como en valor porcentual (es corriente hoy en día que el software represente más del 50% del costo total de un proyecto de sistemas).

La aceptabilidad de los productos de software es de vasto alcance. Su evaluación se basa en el grado de calidad presente tanto en aspectos de productividad como en características del método de desarrollo y del producto mismo.

La tarea de control de calidad no deja de representar una inversión considerable de tiempo y dinero. Sin embargo, los resultados esperados pagan con creces la inversión.

La expectativa es que el esfuerzo extra redunde en:

- una mayor satisfacción del usuario,
- una disminución considerable en la cantidad de errores presentes en el producto,
- una reducción en los costos de mantenimiento debidos a errores,
- un aumento en la confianza depositada en el equipo de desarrollo, y
- mayor facilidad en la administración y generación de nuevas versiones.

En este informe, luego de presentar algunos antecedentes sobre el concepto de calidad del software, se discuten las diferentes modalidaes o enfoques aplicables al control de calidad de software.

Luego se define el concepto de calidad de software, se enumeran las medidas de calidad y se presenta una definición de aceptabilidad en términos de la calidad del software.

Se concluye con recomendaciones generales para los miembros del Club de Investigación Tecnológica. Se incluye además un ejemplo de las medidas y los criterios de aceptabilidad que podría utilizar la compañía ACME para controlar la calidad de un sistema de software.

Se desprende de este informe que para poder disfrutar de los beneficios que ofrece el software de calidad, es necesario: primero, decidir la modalidad de control de calidad a emplear. Segundo, decidir las medidas de calidad que se emplearán en cada punto de control. Finalmente decidir los criterios de aceptabilidad que se emplerán en cada punto de control.

Se debe enfatizar que el costo de los programas de control de calidad no tiene (ni debe) ser oneroso. En muchas ocaciones, el sólo ejercicio de externalizar los criterios de

aceptabilidad (en términos de calidad) puede ser suficiente para mejorar la calidad del software producido.

En organizaciones de cierto tamaño, la función de control de calidad de software debe necesariamente contar con personal y presupuesto propio.

A los beneficios enumerados anteriormente se debería agregar la disminición del riesgo - estamos convencidos que muchos (sino todos) de los projectos que han fracasado recientemente en Costa Rica, no hubieran fracasado si hubiera existido la función de control de calidad en el proyecto.

Agradecimientos

Se agradece la colaboración de Ana Calderón Ureña y Estrellita Vargas Miranda, de Creaciones Digitales CREADISA S.A., por su ayuda en la recopilación de información y edición del texto, respectivamente.

Además se desea expresar el agradecimiento a los miembros del Comité Editorial: Lic. Jorge Walter Bolaños y Dr. Roberto Sasso por las valiosas sugerencias para mejorar la calidad del trabajo, y en especial al Presidente del Club por la confianza depositada.

Del Autor

Ulises Agüero, Director del Programa de Maestría en Computación del Instituto Tecnológico de Costa Rica, es doctor en Ciencias de la Computación por la Universidad del Suroeste de Luisiana, Luisiana USA.

El Dr. Agüero es además, socio y director de Creaciones Digitales CREADISA S.A, fue investigador visitante en el Departamento de Sistemas de la Universidad de Kobe, Kobe, Japón y cuenta con numerosas publicaciones a su nombre, incluyendo una (en 1987) en "Communications of the ACM".

Contenido Página

I Introducción

En la vida cotidiana, las personas evalúan subjetiva u objetivamente los productos que usan y consumen. Del resultado de esta evaluación se generan una serie de apreciaciones que representan características determinantes para la aceptación o rechazo del producto. Sin embargo, debido a la misma naturaleza humana, las apreciaciones subjetivas sobre un producto pueden variar significativamente de un individuo a otro: lo que para uno es una maravilla para otro es el peor de los fracasos.

Conscientes de la situación descrita en el párrafo anterior, las sociedades establecen y renuevan normas de calidad de producción tratando de satisfacer el criterio de las mayorías. La espectativa es que tales normas garanticen el mejoramiento continuo de las condiciones de vida.

Como complemento a tales normas generales, cada individuo asocia con un producto sus propias opiniones. En consecuencia, la **aceptabilidad** de un producto podría decirse que depende de la fusión de normas generales con opiniones personales, según se ilustra en la Figura 1.[1]

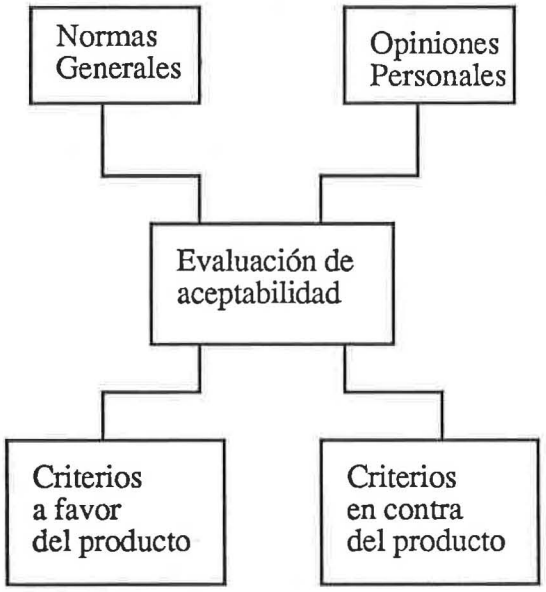


Figura 1.
Determinación de la Aceptabilidad de un Producto

Los productos de software no escapan de la evaluación de aceptabilidad a la que son sometidos los demás tipos de productos. Al respecto, es claro que al menos la opinión del usuario final es un factor determinante para aceptar el software.

En este trabajo se ahonda en los asuntos relacionados con la determinación de la aceptabilidad de un producto de software.

[1] Como es costumbre en escritos tecnicos, el lector debe interpretar que cuando no se citen fuentes es porque lo escrito es original del autor.

Para esto, primero se describen algunos antecedentes del notorio interés actual en aspectos de **aceptabilidad**. Segundo, se propone una clasificación de enfoques de control de calidad. Tercero, se explica el concepto de aceptabilidad del software en términos de la noción de calidad. Se incluye además un ejemplo en el Anexo.

Finalmente, se presentan recomendaciones para la evaluación y control de calidad en nuestro medio.

Antecedentes

Los primeros modelos creados para evaluar las características de un producto de software estuvieron basados en técnicas probabilísticas [MOHAN79]. En este sentido se definen medidas cuantitativas que le dan objetividad a la evaluación. Principalmente se miden aspectos de rendimiento.

En los años siguientes, el énfasis se puso no tanto en la definición de nuevas medidas como en el perfeccionamiento de métodos de medición para las medidas cuantitativas antes desarrolladas. El objetivo en este caso es medir lo más posible en el menor tiempo [MILLE84] [BEIZE84].

Pronto, el interés comenzó a orientarse hacia un enfoque más sistémico de la evaluación de productos de software. Al respecto, las medidas objetivas dan cabida a medidas más subjetivas para darle más significado a la evaluación, involucrando al usuario final como actor importante [KEARN86] [CHARE86].

Al mismo tiempo, se desencadena un notable esfuerzo para que los métodos de desarrollo de software incorporen directamente procesos de evaluación [MILLS87] [SELBY87] [ROPER87].

La certeza sobre la importancia de evaluar detalladamente los productos de software se torna patente cuando autores como [BAYRO87] determinan que existe una demanda creciente por software más complejo para satisfacer necesidades más sofisticadas de información, aunada a un aumento en las exigencias del usuario final con respecto a la forma misma en que se satisfacen sus necesidades.

En este punto, la determinación de la aceptabilidad de un producto de software se torna una disciplina de vastos alcances.

II Modalidades de Control de Calidad

Las medidas de calidad mencionadas anteriormente constituyen las metas de calidad deseables que se pretende tenga el producto final. Pero, ¿cómo asegurar que se logre el nivel de calidad buscado?

El enfoque de control de calidad seguido varía de empresa a empresa. Sin embargo, es raro pensar en un proyecto que no incorpore, de una u otra manera, el control de calidad. En las secciones siguientes se describen algunas posibles modalidades al respecto, las cuales no son excluyentes entre sí. (Sólo las primeras tres no son aporte del autor.)

1. Al Final del Proceso de Desarrollo. [CAVAN87]

Con el fin de no invertir recursos extra (que son escasos) para realizar una función de calidad, muchas veces esta función queda limitada a pruebas de producto terminado para conocer el nivel de calidad específico con que se produjo el software y tomar alguna medida correctiva en caso necesario. Aquí se realizan pruebas de correctitud de programas y de las interfaces, se hacen revisiones funcionales para mostrar que el programa genera los resultados esperados y quizá hasta se realicen algunas pruebas a la seguridad del sistema. Este enfoque podría resumirse en producir software y esperar que la suerte premie al equipo de desarrollo con un producto de alta calidad.

El problema con este enfoque es que se realiza muy tarde en el ciclo de desarrollo como para poder influir en la calidad del producto y, en realidad, hacer sólo inspecciones finales se puede traducir en un desperdicio de recursos para el desarrollo de un producto mediocre que requerirá de recursos adicionales para ser restaurado. La gran ventaja es que disminuye significativamente los costos cuando el producto generado resulta ser aceptable.

2. En Tres Puntos Críticos del Desarrollo. [MILLE84]

Este enfoque propone realizar el control de calidad después de la implementación completa de los módulos individuales, de la integración de dichos módulos en subsistemas y, finalmente, al hacer la integración completa del sistema. De este modo, se pretende tener control sobre las interfaces pues es allí donde se supone que

se generan la mayor cantidad de inconsistencias.

Según los practicantes de este método de control, si al determinar la calidad de los módulos individuales, éstos no satisfacen los estándares, se procederá a corregir los problemas y no se dará la integración hasta tanto no se tenga un módulo de la calidad deseada. Luego, cuando todos y cada uno de los módulos haya pasado las pruebas de calidad, se procederá a la integración en subsistemas. Así, se garantizará que cualquier problema que se genere será consecuencia de la integración misma, y su detección y control serán más rápidos.

Cada módulo es visto como un producto en sí mismo, con simples inspecciones finales como las mencionadas anteriormente. El mayor problema con este enfoque es que no se está dando importancia a las fases iniciales del ciclo de vida del software. Al hacerlo se trabaja a menudo sin especificaciones a priori de la calidad esperada al término de cada fase.

El ciclo de vida del software es un proceso delicado de desarrollo en donde intervienen humanos quienes anteponen criterios personales y profesionales al hacer su trabajo. La posibilidad de cometer errores en cada fase es alta y dejar el control hasta el final de todo el proceso es riesgoso y no permite diseñar, sólo calificar, la calidad de lo que se produce.

La ventaja de este enfoque sobre el anterior es que, con un buen particionamiento del sistema en módulos se puede lograr, con respecto a control de calidad, una cobertura aceptable del ciclo de vida del software.

3. Durante Todo el Proceso de Desarrollo. [CAVAN87]

Control completo es el secreto para obtener un producto de alta calidad. Esto significa que se debe seguir una fase de verificación a todo lo largo del ciclo de vida del software, empezando en la especificación de requerimientos hasta la integración y validación finales. No se puede construir un producto de alta calidad a no ser que celosamente ésta, la calidad, se cuide y se mejore en cada fase diferente por la que atraviesa el producto desde sus orígenes.

¿Por qué se requiere de un control así tan riguroso? Porque en cada fase se pueden dar errores de omisión, concepto o técnicos que pueden ser arrastrados hasta el producto final donde serían más difíciles de detectar.

Es necesario, entonces, verificar que durante cada fase del ciclo de vida del software y al final de la misma se satisfagan todos los requerimientos y expectativas funcionales y de calidad esperados.

De esta forma, no se pasará a la siguiente fase a menos que el producto generado por la fase actual sea un producto de alta calidad.

Es más fácil detectar y/o corregir un error en la fase en que éste se produce, que esperar hasta que el producto final haya salido para hacerlo. Por supuesto, las fases en que se efectúan revisiones dependen del método de desarrollo utilizado. De hecho, una empresa que decide mejorar la calidad de su software debe reconsiderar las fases de su método de desarrollo, de tal forma que incorporen explícitamente los puntos de revisión, las medidas de calidad pertinentes, los estándares de cada medida, y la forma en que se efectuarán las mediciones.

Este enfoque pretende entonces hacer uso de revisiones técnicas e inspecciones del producto no sólo en ciertas fases finales del desarrollo (lo cual dejaría sin control las fases en que se genera el producto) sino en cada fase claramente diferenciada del desarrollo del software. Haciendo un control de cobertura total del ciclo de desarrollo se podrá ver la evolución del producto en paralelo a la evolución de la calidad del mismo, y se podrá saber con certeza el nivel de calidad del producto en un momento dado.

Revisiones Técnicas

Una revisión técnica formal del software es una actividad de control de calidad que permite detectar errores funcionales, de lógica, o de implantación, verificar la satisfacción de los requerimientos establecidos y controlar la uniformidad en el método de desarrollo usado y el cumplimiento de los estándares.

Las revisiones técnicas son realizadas en grupos de dos a cinco personas, dependiendo de la complejidad del proyecto, una de las cuales es el productor del elemento de software sometido a revisión. La idea es someter el producto a crítica técnica para determinar la calidad del mismo en términos de las medidas preestablecidas, detectar sus errores, proponer mejoras y determinar el progreso general en el plan del proyecto. De estas revisiones técnicas salen reportes de control que se distribuirán entre los involucrados para tomar las medidas del caso. Además, es durante la revisión técnica que se hacen las mediciones de las características de calidad. A ningún producto se le da el visto bueno para continuar en la siguiente fase a no ser que el equipo de revisión esté satisfecho con el mismo.

Ventajas y desventajas

Este enfoque de control sobre todo el ciclo de desarrollo trae la ventaja de que permitirá al equipo de trabajo decidir sobre el nivel de calidad deseado (o diseñado) y obtener dicho nivel de calidad en el producto final.

Otros autores comparten la importancia del control de calidad continuo:

La detección temprana de errores es mucho más efectiva que esperar hasta

la etapa de aceptación y luego reefectuar el trabajo. [2] **[POSTO87]**

Entre más temprano se detecten o anticipen los problemas de calidad en el proceso de desarrollo del software, más efectivas son las medidas correctivas. [3] **[BAYRO87]**

La desventaja del enfoque es que puede aumentar significativamente el costo inicial del desarrollo y demanda una gran disciplina y organización.

4. El Enfoque Individualista.

Es frecuente que cada desarrollador defina subjetivamente las medidas de calidad para las tareas que le corresponden y además las verifica por sí mismo. Si existen estándares documentados, su cumplimiento se deja a criterio del desarrollador.

A menudo el control es desordenado, favorece los abusos de confianza, y se centra básicamente en los requerimientos funcionales (entradas/salidas) del problema. A este nivel, la calidad técnica del software no es medida por segundos, y depende extensamente de las características propias del desarrollador.

La ventaja del enfoque reside en su eficacia cuando los desarrolladores son personas de gran experiencia y capacidad. Aunque este tipo de personas a menudo demandan altos honorarios, los costos generales pueden ser relativamente bajos debido a la simplicidad administrativa del esquema.

5. El Enfoque Orientado al Usuario Final

La calidad del proceso de desarrollo y del producto es medida sólo en términos de la satisfacción del usuario final. Los métodos de desarrollo que se basan en este enfoque involucran una serie de puntos de aprobación, donde grupos de usuarios examinan los documentos generados y los aprueban después de interactuar con los desarrolladores.

Una característica importante de esta modalidad es que no interesa tanto si el producto es lo mejor posible; lo importante es que el usuario esté satisfecho.

Una ventaja es que sensibiliza al usuario final con respecto a la problemática del proceso de desarrollo. Mas la desventaja es que puede provocar descuidos en la calidad técnica del producto. Los costos asociados al enfoque dependen en gran medida de la actitud que asuman los usuarios mismos. Si son

[2] Early error detection is much more efective than waiting for testing and then having to do rework [Posto87].

[3] The earlier in the software process that quality problems are detected or anticipated, the more efectivie the counter measures are [Bayro87]

usuarios que desde un inicio se incorporan al proceso de desarrollo, los costos serán mucho más bajos que cuando los usuarios comienzan a plantear sus objeciones e inquietudes al final del proceso. Esto pues las modificaciones extemporáneas son bastante costosas en tiempo (y dinero).

6. El Enfoque Orientado a los Procedimientos

Se establecen estándares extensos sobre los pasos a seguir durante el proceso de desarrollo y sobre los documentos que debe generar cada etapa. El cumplimiento de tales estándares es verificado con diferentes niveles de exhaustividad, dependiendo de la importancia y organización del proyecto, y del presupuesto asignado.

En tal situación, la calidad del software mismo no es medida, bajo el supuesto que un buen método es suficiente para crear un buen producto.

Una ventaja del enfoque es que sistematiza el proceso. Además, el costo asociado al enfoque se reduce a la inversión en el desarrollo de un buen manual de procedimientos. Tiene la desventaja de que tiende a no discriminar entre la muy variada gama de problemas que a menudo se presentan en una empresa.

7. El Enfoque Orientado al Producto

Se basa en el control de la calidad técnica del producto. Se establecen estándares de codificación, modulación, interfaz persona-máquina, eficiencia del código, y otros. En el peor de los casos, los estándares no son verificados por razones económicas o humanas. En el mejor de los casos, la verificación es detallada por otra persona que no es el desarrollador.

Una ventaja principal es que propicia un fácil mantenimiento del software y el uso posterior de los módulos creados. Sin embargo, tiende a ignorar las preferencias del usuario final, las cuales pueden convertirse en factores determinantes para la aceptación del producto.

El costo depende significativamente del detalle con que se especifican los estándares, pues conforme aumenta el detalle crece el tiempo requerido para el desarrollo y control.

8. El Enfoque Jerárquico

Se considera aquí que debe existir una jerarquía de control de calidad bien definida. La función de controlar es reconocida explícitamente como parte íntegra de todo proyecto y distribuída en varios niveles de la organización. Los productos y procedimientos de un nivel son controlados

por el nivel superior, con base en criterios y medidas de calidad claramente especificados.

Por supuesto, este enfoque de control de calidad es el que representa la mayor complejidad, pero a la vez minimiza la presencia de errores y el esfuerzo requerido para modificar el producto.

El costo del desarrollo inicial puede ser alto, pero se compensa con una reducción en los costos por corrección de errores y modificaciones.

III Aceptabilidad en Términos de Calidad

En esta sección primero se define calidad tratando de ajustarse a lo sugerido en la literatura consultada. Segundo, se caracteriza un compendio de medidas de calidad obtenido de la literatura. Tercero, se brinda una lista (propia) de factores relacionados con mediciones. Finalmente, se define aceptabilidad de un producto de software en el contexto de medidas de calidad.

1. Concepto de Calidad del Software.

Desde un inicio (ver [MOHAN79]), el término ***calidad*** fue utilizado para referirse al conjunto de características, deseables o no, presentes en un producto de software o en el proceso que lo genera.

En términos generales, calidad se refiere a los siguientes tres aspectos [BAYRO87] (ver Figura 2):

Calidad con respecto a Productividad
Se refiere al tiempo y el costo de producción.

Calidad del Método de Desarrollo.
Se refiere a la naturaleza de los procedimientos seguidos durante el diseño y construcción del software.

Calidad del Producto.
Se refiere a las características del producto de software, generadas con respecto a un método dado y bajo ciertas restricciones de productividad.

Figura 2
Tres Tipos de Calidad

De acuerdo con Pressman, calidad del software se puede definir como

conformidad con requerimientos funcionales y de rendimiento

explícitamente descritos, con estándares de desarrollo explícitamente documentados, y con características implícitas esperadas de todo software desarrollado profesionalmente.[4] **[PRESS87]**

Cabe aclarar sobre la definición que los estándares de desarrollo se refieren tanto a aspectos de procedimiento como a características técnicas del software.

Así, al hablar de calidad del software se debería siempre pensar en la importancia de tener una definición completa no sólo de los requerimientos de productividad de la empresa al desarrollar software sino, además, de los requerimientos de **calidad** que satisfagan las expectativas del usuario.

Debería existir, también, una definición de los estándares de calidad que el software debe satisfacer. Esto es, las características de interés, las relaciones y prioridades entre ellas y, además, el grado al cual éstas deben estar presentes en el producto final [BAYRO87].

Esta definición de estándares permitirá la evaluación consistente del software, ya que son los estándares los que darán la base de comparación para determinar la calidad del software producido en un momento dado.

Finalmente, no se debería restar importancia a aquellos requerimientos de calidad que por considerarse obvios quizá no se mencionan ni documentan pero que son igualmente importantes para determinar la calidad de las aplicaciones de software producidas. Así, la definición de calidad de software particular a cada empresa variará dependiendo de los intereses y prioridades establecidos a la luz de las posibilidades de la empresa y los niveles de calidad requeridos por los usuarios.

Definir la calidad del software producido por una empresa supone, entonces, la especificación de un conjunto de características de calidad deseables en el producto y el establecimiento de los niveles de calidad aceptables. Esto último se hace definiendo, para cada característica de calidad, un conjunto de factores (y su nivel deseable) que son susceptibles de ser medidos en alguna forma y cuya medición permite determinar el nivel de calidad obtenido en dicha característica [KISHI87].

2. Medidas de Calidad.

Una posible forma de modelar calidad es por medio de una jerarquía de medidas que cubran los tres tipos de calidad mencionados en la Figura 2. Cada nivel de la jerarquía representa un conjunto de medidas con cierto grado de abstracción, tal que cada medida de

[4] Conformance to explicitly stated functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software [press87]

un nivel dado es definida en términos de medidas menos abstractas. La Figura 3 muestra un esquema general de una jerarquía de dos niveles.

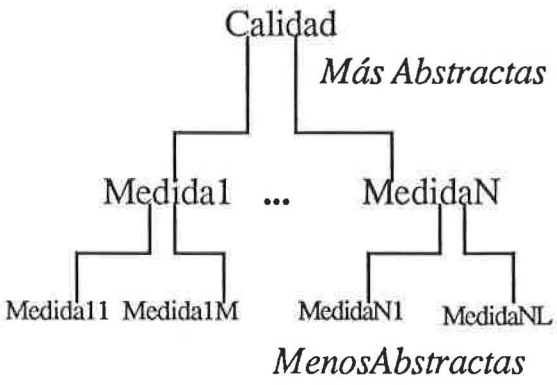


Figura 3
Jerarquía de Medidas

En la literatura consultada, la mayor parte de los trabajos se refiere a sólo uno de los tres tipos de calidad mencionados: **calidad del producto**. Las medidas de los otros dos tipos (productividad y método) son más intuitivas. La medida de productividad: costo y tiempo, no requiere explicación, mientras que las medidas del método deben forzosamente limitarse a medir los productos (o subproductos) del método utilizado: documentación técnica y de usuario

Con respecto a las medidas de la calidad del producto, un primer nivel de medidas podría incluir las siguientes [KEARN86] [PRESS87] [CHARE86]:

- **Confiabilidad:**
Grado en que el programa opera satisfactoriamente (sin errores de ejecución).

- **Correctitud:**
Grado en que el programa satisface las especificaciones originales y los objetivos del usuario.

- **Eficiencia:**
Cantidad de recursos y código requeridos para que el programa realice su función.

- **Flexibilidad:**
Cantidad de esfuerzo necesario para modificar el programa.

- **Integridad:**
Grado de control sobre el acceso al software y a los datos por parte de usuarios no autorizados.

- **Integración:**
Cantidad de esfuerzo necesario para integrar un sistema con otro; facilidad para crear interfaces de comunicación entre sistemas.

- **Usabilidad:**
Cantidad de esfuerzo requerido para aprender y operar correctamente el sistema.

- **Mantenimiento:**
Cantidad de esfuerzo requerido para entender, corregir, adaptar y mejorar el software.

- **Portabilidad:**

Facilidad con la cual un programa puede ser transferido de un ambiente computacional a otro y continuar trabajando en óptimas condiciones.

- **Reuso:**
Habilidad de un elemento del software para ser usado otra vez en alguna otra aplicación.

- **Separación:**
Grado de separación entre el código que implanta el diálogo persona-máquina y el código que efectúa otras operaciones que no involucran al usuario directamente.

Un segundo nivel de medidas (menos abstracto) incluiría las siguientes [KEARN86] [PRESS87] [CHARE86]:

- **Auditoría:**
Facilidad para determinar el grado de satisfacción de los estándares.

- **Precisión:**
Grado de precisión en los cálculos y el control.

- **Comunicaciones:**
Uso de estándares para establecer protocolos y crear rutinas de interface entre módulos.

- **Estructuras:**
Uso de estándares en la representación de datos a través de todo el programa.

- **Completitud:**
Grado en el cual las funciones requeridas han sido completamente implementadas.

- **Consistencia:**
Uniformidad de las técnicas de diseño e implantación usadas durante el desarrollo del software.

- **Tamaño:**
Grado en que una función fue implementada usando una cantidad mínima de código.

- **Errores:**
Grado en el cual se garantiza la continuidad de la operación luego de la aparición de un error (i.e., capacidad de recuperación).

- **Eficiencia:**
Medida del tiempo de procesamiento y espacio de memoria requerido para la operación del programa.

- **Ampliación:**
Facilidad con que los requerimientos de memoria, arquitectura y/o diseño de procedimientos pueden ser extendidos para atender a nuevas necesidades.

- **Generalidad:**
Alcance de las aplicaciones potenciales de los diferentes componentes del programa.

- **Hardware:**
Grado de dependencia del software de las características físicas del sistema (hardware).

- **Modularidad:**

Grado de dependencia funcional entre los módulos que componen el sistema.

- Operación:
Facilidad de operación del software.

- Seguridad:
Disponibilidad de mecanismos de control y/o protección del software y los datos.

- Autodocumentación:
Uso de documentación incluída como parte del código para la explicación de detalles de implantación.

- Simplicidad:
Grado con el cual un programa puede ser entendido sin dificultad.

- Independencia:
Grado de independencia del software desarrollado del software de sistemas (por ejemplo, Sistema Operativo, Utilitarios, y Rutinas de Entrada y Salida).

- Seguimiento:
Habilidad para hacer un seguimiento desde la implantación o el diseño hasta la especificación de requerimientos.

- Entrenamiento:
Grado con que el software ayuda a familiarizar al nuevo usuario con la operación del mismo.

Cada medida del primer nivel puede asociarse a varias medidas del segundo nivel.[5] La Figura 4 muestra un ejemplo. (Los interesados en una cobertura más amplia entre medidas de diferentes niveles pueden, por ejemplo, consultar [PRESS87], página 437, o [CHARE86], página 237.)

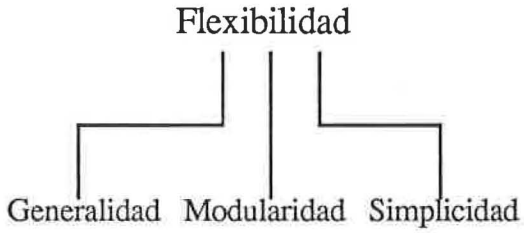


Figura 4
Relaciones entre Medidas

3. Mediciones de las Medidas

Definidas las medidas, el siguiente paso es determinar la manera en que se realizarán las mediciones de calidad. Estas pueden ser cuantitativas o cualitativas.

Entre más mediciones cuantitativas de una característica se usen, más objetivo será el resultado obtenido, ya que las cualitativas son expresiones de conformidad basadas en consideraciones (o puntos de vista)

[5] Por supuesto, la definición del nivel de abastracción de una medida es bastante subjetiva.

individuales o de grupo que pueden ser desviadas de la realidad en forma circunstancial.

La manera en que se efectúa una medición depende en gran escala de los siguientes factores:

- Las características de los métodos de desarrollo y mantenimiento empleados.
- Las herramientas de desarrollo y control de calidad que apoyan la labor del ingeniero de software.
- La importancia de la medida en la aplicación de interés.
- El presupuesto asignado al proyecto.
- La naturaleza de la medida y la aplicación misma.
- La organización de los participantes en el proyecto.
- La experiencia del grupo de control de calidad.
- La claridad y la especificidad de los documentos que definen los estándares.

Con cada medición es aconsejable fijar el estándar de la medida. Esto se puede realizar utilizando información histórica, si la hay, o bien, haciendo uso de información obtenida de un análisis de los niveles de calidad deseables en el software producido.

Los estándares deben ser cuidadosamente fijados para que no sean ni muy fáciles de lograr ni inalcanzables, pues en ambos casos provocarían desmotivación en el equipo de desarrollo.

Finalmente, tómese en cuenta que existen ciertas formas cuantificables de medición, vastamente estudiadas en la literatura, y que pueden usarse en muchas situaciones. Por ejemplo, pueden efectuarse mediciones relacionadas al tamaño del código, requerimientos de espacio y tiempo, y rendimiento de la base de datos.

La discusión detallada de la cuantificación de las medidas está más allá de los objetivos del presente informe. Los interesados pueden consultar la bibliografía al respecto: por ejemplo, [BAYRO87], [GRADY87], [MILLE84], [MOHAN79], y [ROPER87].

4. Aceptabilidad del Software

Tradicionalmente, la aceptabilidad del software se define únicamente en términos de si satisface o no los estándares fijados.

Seguidamente se presenta una alternativa que permite considerar de una forma global todos los criterios de aceptabilidad de sistemas para una organización.

Considerese calidad como conformada por los tres tipos de la Figura 2. Al respecto, es útil definir *aceptabilidad* como *una*

estimación del grado al que los estándares de las medidas se han alcanzado. Tal estimación se expresa asignándole al software un **estado de aceptabilidad** de acuerdo con las siguientes definiciones:

dudoso: las mediciones muestran que la mayoría de los estándares no se alcanzaron.

insatisfactorio: las mediciones muestran que ningún estándar se alcanzó

aceptable: las mediciones muestran que la mayoría de los estándares se alcanzaron.

satisfactorio: las mediciones muestran que todos los estándares se alcanzaron.

Notese que los estados de aceptabilidad son tales que cualquier producto de software que califica para el estado **insatisfactorio** también califica para el estado **dudoso** y cualquier producto que califica para el estado **satisfactorio** también califica para el estado **aceptable**. La Figura 5 muestra la relación entre estados.

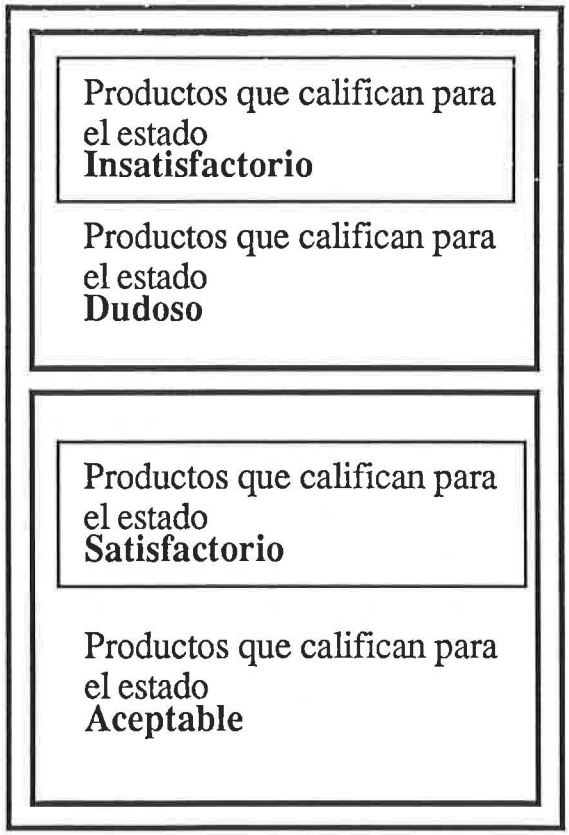


Figura 5
El Universo de Aceptabilidad

Un enfoque de este tipo puede ser de mucha utilidad cuando lo que se desea es una determinación global sobre un sistema de software particular.

Es posible desarrollar un argumento basado en apreciaciones sobre aspectos particulares de un sistema, determinar el estado de aceptabilidad de cada aspecto y deducir el estado de aceptabilidad del sistema a partir de los estados particulares.[6]

[6] Esta sección es una adaptación parcial de la teoría de plausibilidad. [Agüer87] [Agüer89].

Tanto para el desarrollo del argumento, como para deducir automáticamente el estado de aceptabilidad del sistema a partir de los estados particulares (en varios niveles) es conveniente contar con herramientas de software apropiadas.

Para esto es necesario que las herramientas permitan modelar el argumento en términos de distintos aspectos interrelacionados.[7]
En el Anexo se presenta un ejemplo del modelo de un argumento de aceptabilidad de un sistema que se planteó utilizando el software Plausible-CADM.

[7] El Ing. Eduardo Carvajal, el autor y Creaciones Digitales s.a. están elaborando un programa denominado Plausible-CADM (Computer-Aided Decision-Making) que eficazmente permite desarrollar argumentos de aceptabilidad.

IV Conclusiones

La literatura consultada demuestra que las discusiones sistémicas sobre la naturaleza de la aceptabilidad y calidad del software apenas comienzan. Sin embargo, esto no le resta relevancia a la materia, pues los productos de software, en muchos sentidos, no difieren de otros productos más convencionales, especialmente con respecto a la satisfacción del usuario final. Por lo tanto, se hacen las siguientes recomendaciones generales a los miembros del Club de Investigación Tecnológica:

-.El tema de la aceptabilidad (de la calidad) del software debería discutirse a nivel de empresa para determinar el papel que tiene o debe tener en la misma. Para esto podrían organizarse sesiones de discusión donde participen representantes de la gerencia, usuarios finales y desarrolladores. Comisiones específicas decidirían sobre los componentes de los niveles de medidas, estándares y mediciones. A la vez, se haría una estimación de la inversión requerida.

-.La adquisión de equipo debe ir acompañada de una evaluación de las herramientas disponibles para las mediciones de calidad. Los proveedores nacionales de hardware y software consultados coinciden en que existe un desinterés marcado, de parte de las empresas, en considerar herramientas de desarrollo orientadas al control de calidad.

-.Debería existir al menos un grupo externo al grupo de desarrollo que evalúe la aceptabilidad del software. Por supuesto, dentro del grupo de desarrollo mismo pueden existir otros dedicados a la misma tarea.

-.El desarrollo y control de estándares de medición debería ser una actividad permanente dentro de la empresa.

-.Sería adecuado que los niveles altos de la administración de la empresa participen en la definición de medidas de calidad para asegurar una relación íntima con las políticas gerenciales y no solo con la visión de los desarrolladores.

-.Como en muchas actividades empresariales, para obtener beneficios del control de calidad hay que invertir en éste.

Una forma de empezar es organizando el esquema de desarrollo de software. Cada proyecto podría constar de los siguientes componentes humanos:

El responsable del proyecto. Es el encargado de dirigir el desarrollo del sistema de software y de coordinar con el grupo de control de calidad.

Entre otras cosas, el responsable del proyecto evacúa dudas de los desarrolladores a su cargo, con respecto a las especificaciones que se les entregan.

Además, estima el tiempo requerido por cada tarea asignada a un desarrollador, de tal manera que el grupo de control pueda darle seguimiento.

También negocia con sus superiores la asignación de recursos para su proyecto.

Finalmente, se encarga de controlar las diferentes versiones del sistema que se está desarrollando, para asegurar consistencia entre las tareas asignadas.

El grupo de control de calidad. Su tamaño depende del tamaño del proyecto. Vela por el cumplimiento de todos los estándares y por que los programas o módulos se adhieran a las especificaciones funcionales y a las medidas de calidad.

Además, controlan el tiempo de desarrollo con respecto a las estimaciones y producen un reporte final sobre el proyecto. Este reporte debe incluir una determinación del costo real del proyecto, una evaluación de la conveniencia de los estándares y medidas utilizados, sugerencias sobre mejoras al esquema de control de calidad, y la documentación sobre el control efectuado.

Los desarrolladores. La cantidad de desarrolladores involucrados varía durante el proyecto, según la etapa de desarrollo en que se encuentre.

Su labor debe basarse en especificaciones claras y por escrito sobre cada tarea. Además, el resultado del control de calidad de su trabajo debe comunicárseles por escrito para que efectúen las correcciones del caso. Todos deben contar con documentos precisos sobre los estándares que deben seguir.

Para concluir, debe quedar claro que para que un esquema como el descrito funcione, se requiere una inversión significativa en el diseño de manuales de estándares y de formularios para revisiones de calidad, control de tiempos y redacción de especificaciones.

Bibliogrfía

[ACKER89] Ackerman, A.Frank, Buchwald, Lynne S. y Lewski, Frank H. "Software Inspections: An Effective Verification Process." *IEEE Software*. May 1989.

[AGUER87] Agüero, U. y Dasgupta, S. "A Plausibility-Driven Approach to Computer Architecture Design." *Communications of the ACM*. Vol. 30, No. 11, November 1987.

[AGUER88] Agüero, U. "La Plausibilidad de los Diseños de Computadoras." *Tecnología en Marcha*. Noviembre 1988.

[BAYRO87] Basili, Victor R. y Rombach, H.Dieter. "Implementing Quantitative SQA: A Practical Model." *IEEE Software*. September 1987.

[BAYSEL87] Basili, Victor R. y Selby, Richard W. "Comparing the Effectiveness of Software Testing Strategies." *IEEE Transactions on Software Engineering*. Vol.SE-13, No.12, Semtember 1987.

[BEIZE84] Beizer, Boris, *Software System Testing and Quality Assurance*, Van Nostrand, 1984.

[CAVAN87] Cavano, Joseph P. y LaMonica, Frank S. "Quality Assurance in Future Development Environments." *IEEE Software*. September 1987.

[COLLO87] Collofello, James S. y Buck, Jeffrey J. "Software Quality Assurance for Maintenance." *IEEE Software*. September 1987.

[CHARE86] Charette, Robert N. *Software Engineering Environments: Concepts and Technology*. Intertext Publications, Inc. New York, 1986. pp.231-260.

[DROME88] Dromey, R. Geoff. "Systematic Program Development." *IEEE Transactions on Software Engineering*. Vol.14, No.1, January 1988.

[GRADY87] Grady, Robert B. "Measuring and Managing Software Maintenance." *IEEE Software*. September 1987.

[KEARN86] Kearney, Joseph K., Sedlmeyer, Robert, Thompson, William y otros. "Software Complexity Measurement." *Communications of the ACM*. Vol.29, No.11, November 1986.

[KISHI87] Kishida, Kouichi, Teramoto, Masanori, Torii, Koji y Urano, Yoshiyori. "Quality Assurance Technology in Japan." *IEEE Software*. September 1987.

[MILLE84] Miller Jr., Edward F. "Software Testing Technology: An Overview." *Handbook of Software Engineering*.

Cap. 16. pp 359-379.

[MILLS87] Mills, Harlan D., Dyer, Michael, y Linger,Richard C. "Cleanroom Software Engineering." *IEEE Software*. September 1987.

[MOHAN79] Mohanty, Siba N. 'Models and Measurements for Quality Assessment of Software." *Computing Surveys*. Vol.11, No. 3, September 1979.

[POSTO87] Poston, Robert M. y Bruen, Mark W. "Counting Down to Zero Software Failures." *IEEE Software*. September 1987.

[PRESS87]. Pressman, Roger S. *Software Engineering: A Practitioner's Approach*. Second Edition, McGraw-Hill Book Company,USA. 1987, pp.433-463.

[ROPER87] Roper, M. y Smith, P. "A Structural Testing Method for JSP Designed Programs." *Software: Practice and Experience*. Vol.17, No.2 , February 1987.

[SELBY87] Selby, Richard W., Basili, Victor R. y Baker, F.Terry. "Cleanroom Software Development: An Empirical Evaluation." *IEEE Transactions on Software Engineering*. Vol. SE-13, No. 9, September 1987.

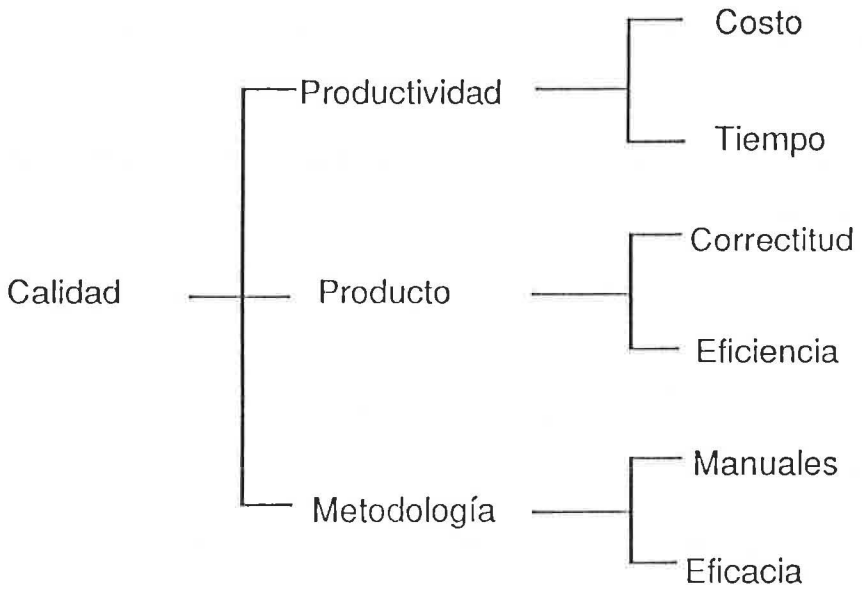
[URIAS88] Urías Muños, Carlos. "An Approach to Software Product Testing." *IEEE Transactions on Software* Engineering. Vol.14, No.11, November 1988.

[WALLA89] Wallace, Dolores R. y Fujii, Roger U. "Software Verification and Validation: An Overview." *IEEE Software*. May 1989.

Anexo: Ejemplo de la Compañía ACME

Suponga un caso hipotético y general de determinación de la aceptabilidad de un sistema de software.

Un primer paso podría ser la definición de una estructura jerárquica, de tres niveles en este caso, para interrelacionar las medidas de interés, según se muestra en la siguiente figura:



Según la sección sobre aceptabilidad de software, la idea sería determinar si la mayoría de los estándares se cumplen o no para las medidas seleccionadas.

Para esto, lo más obvio sería proceder de las medidas más específicas hacia las más generales (por ejemplo, seguridad es más específica que producto). En cada paso se efectuaría la medición de la medida para evaluar el nivel al que se satisfacen los estándares. Esta medición produciría un estado de aceptabilidad para la medida.

Teniendo el estado de aceptabilidad de cada medida el problema sería ahora inferir el estado de aceptabilidad de las medidas del siguiente nivel, a partir de los estados ya derivados.

Para lograrlo, pueden aplicarse las siguientes leyes de aceptabilidad:

(L1) ACEPTABLE + SATISFACTORIO = ACEPTABLE

(L2) SATISFACTORIO + SATISFACTORIO = SATISFACTORIO

(L3) DUDOSO + SATISFACTORIO = DUDOSO

(L4) ACEPTABLE + DUDOSO = DUDOSO

Con respecto al ejemplo que nos ocupa, se determinarían primero los estados de aceptabilidad de las medidas **costo, tiempo, correctitud, seguridad, manuales y eficiencia.** A partir de los estados de **costo y tiempo** se genera el estado de aceptabilidad de **productividad.** La siguiente tabla resume la forma en que se obtendría el estado de cada medida a partir de los estados de las otras:

Medida	**Estado deducido a partir de**
Calidad	Productividad, Código, Metodología
Productividad	Tiempo, Costo
Tiempo	Evaluación propia
Costo	Evaluación propia
Producto	Correctitud, Seguridad
Correctitud	Evaluación propia
Seguridad	Evaluación propia
Metodología	Manuales, Eficacia
Manuales	Evaluación propia
Eficacia	Evaluación propia

Ahora supóngase que las evaluaciones iniciales produjeron los siguientes resultados:

Tiempo: En el 60% de los casos en que se aplicaron "benchmarks" se cumplió a cabalidad con el estandar mínimo. Esto lleva a asignarle el estado de aceptabilidad ACEPTABLE.

Costo: El sistema se desarrolló dentro del presupuesto establecido. Por lo tanto, se le asigna el estado SATISFACTORIO.

Correctitud: Todas las pruebas realizadas produjeron resultados favorables. Por lo tanto, se le asigna el estado SATISFACTORIO.

Eficiencia: Las mediciones sobre espacio requerido indicaron que se satisfacieron los estándares al respecto en todos los casos. Así se le asigna el estado SATISFACTORIO.

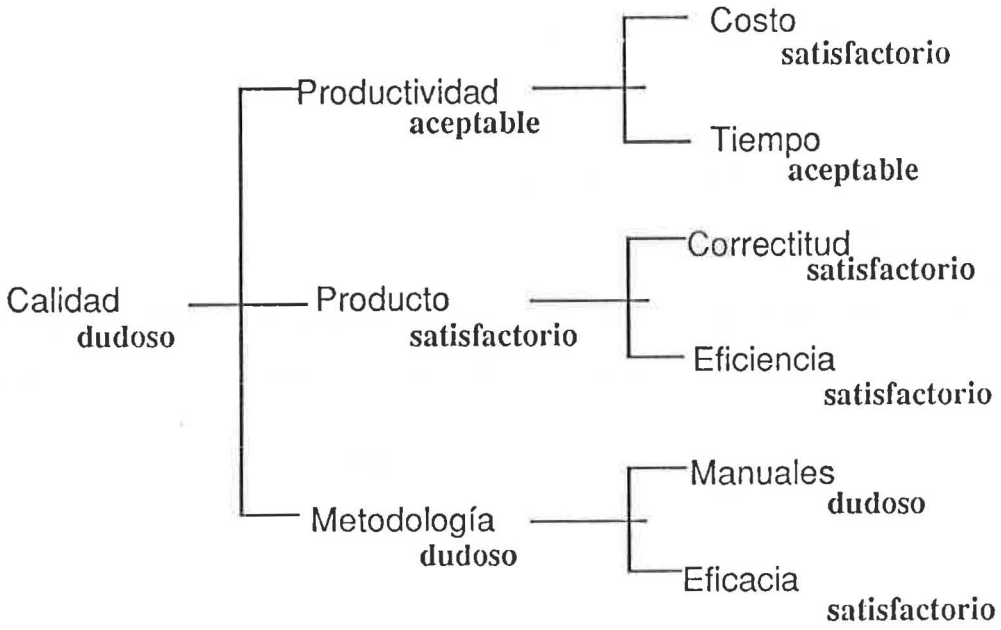
Manuales: Las revisiones de estilo y los documentos de desarrollo mostraron que raramente se siguió lo indicado en los manuales. En consecuencia el estado asignado es DUDOSO.

Eficacia: El procedimiento que se siguió produjo los resultados funcionales esperados. Como resultado se asigna el estado SATISFACTORIO.

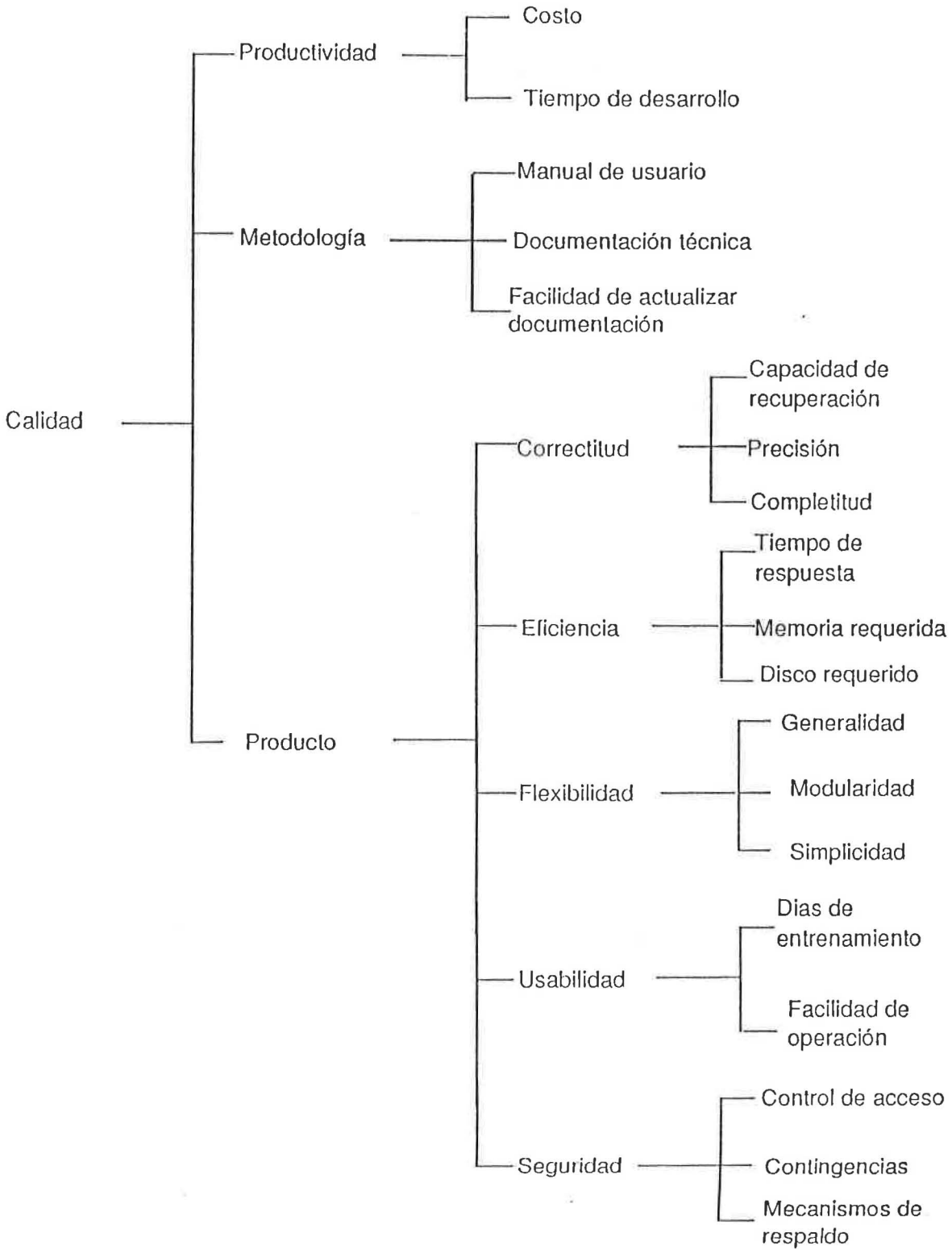
Si se aplican las leyes se obtendrían los siguientes estados para cada medida:

Medida	**Estado**	**deducido a partir de**	**Leyes**
Calidad	DUDOSO	ACEPTABLE SATISFACTORIO DUDOSO	L1 L4
Productividad	ACEPTABLE	ACEPTABLE SATISFACTORIO	L1
Tiempo	ACEPTABLE	Evaluación propia	
Costo	SATISFACTORIO	Evaluación propia	
producto	SATISFACTORIO	SATISFACTORIO SATISFACTORIO	L2
Correctitud	SATISFACTORIO	Evaluación propia	
Seguridad	SATISFACTORIO	Evaluación propia	
Metodología	DUDOSO	DUDODOSO SATISFACTORIO	L3
Manuales	DUDOSO	Evaluación propia	
Eficacia	SATISFACTORIO	Evaluación propia	

En la siguiente figura se puede apreciar la evaluación realizada:



En la siguiente página se presenta una estructura jerárquica más realista para la evaluación de un sistema importante en la Compañía ACME.


Calidad
Productividad
Costo
Tiempo de desarrollo
Metodología
Manual de usuario
Documentación técnica
Facilidad de actualizar documentación
Producto
Correctitud
Capacidad de recuperación
Precisión
Completitud
Eficiencia
Tiempo de respuesta
Memoria requerida
Disco requerido
Flexibilidad
Generalidad
Modularidad
Simplicidad
Usabilidad
Dias de entrenamiento
Facilidad de operación
Seguridad
Control de acceso
Contingencias
Mecanismos de respaldo