8-2010

Utilizando el Modelo de Calidad de McCall y el Estándar ISO-9126 para la Evaluación de la Calidad de Sistemas de Información por los Usuarios

Juan Manuel Gómez Reynoso
Universidad Autónoma de Aguascalientes, jmgr@correo.uaa.mx

Estela Lizbeth Muñoz Andrade
Universidad Autónoma de Aguascalientes, elmunoz@correo.uaa.mx

Jorge Eduardo Macías Díaz
Universidad Autónoma de Aguascalientes, jemacias@correo.uaa.mx

Utilizando el Modelo de Calidad de McCall y el Estándar ISO-9126 para la Evaluación de la Calidad de Sistemas de Información por los Usuarios

Juan Manuel Gómez Reynoso
Universidad Autónoma de Aguascalientes
jmgr@correo.uaa.mx

Estela Lizbeth Muñoz Andrade
Universidad Autónoma de Aguascalientes
elmunoz@correo.uaa.mx

Jorge Eduardo Macías Díaz
Universidad Autónoma de Aguascalientes
jemacias@correo.uaa.mx

ABSTRACT

Los sistemas de información son una herramienta fundamental en la operación diaria de las organizaciones. El software es una objeto totalmente abstracto del cual los usuarios solo perciben su efecto, más no pueden conocer la construcción del mismo. Además, tradicionalmente los usuarios participan solo en etapas iniciales o al final de la construcción de un nuevo sistema. Esta investigación involucró a los usuarios en la evaluación de la calidad de software a través de un caso de estudio donde participaron dos grupos de usuarios diferentes evaluando 2 prototipos del mismo sistema. Se evaluaron mediante cuestionarios construidos basados en el modelo de calidad de McCall (1977) combinado con un estándar (ISO/IEC_9126-1, 2001). Los resultados indican que la participación de los usuarios en la evaluación de la calidad realmente impactan la calidad.

Palabras Clave

Calidad de software, Ingeniería de Software, sistemas de Información, evaluación, usuarios.

INTRODUCCIÓN

La finalidad de todo software es satisfacer las necesidades del usuario. El uso de aplicaciones de software dentro de las organizaciones se ha convertido en una necesidad para el buen funcionamiento y desarrollo de las mismas. Frecuentemente estas organizaciones se enfrentan a la liberación de software que no cumplen totalmente con sus necesidades. Lo cual puede ir desde el incumplimiento de los requerimientos establecidos hasta la facilidad de uso. Debido a esto surge la necesidad de que el software pase por una evaluación detallada con el fin de mejorar la calidad del producto final y satisfacer las necesidades del usuario. El proceso de medición de la calidad del software (CS) debe implementarse en todo el ciclo de vida del mismo, pero es importante destacar la evaluación del producto terminado, ya que el tener un alto nivel de calidad en el proceso de desarrollo, no garantiza productos finales de calidad.

Actualmente, existen diferentes modelos de calidad que indican las cualidades deseables para determinar la calidad de un producto de software. Sin embargo, tales modelos no establecen mecanismos definidos para su evaluación, por lo que la presente investigación propone integrar y definir, en base a estos modelos, los principales factores que se pueden analizar y la forma de evaluarlos, considerando principalmente la integración de los usuarios finales en dicha evaluación, todo esto para contribuir a lograr aplicaciones de software de alta calidad.

MARCO TEÓRICO

Una preocupación muy importante en la industria del software es desarrollar productos con calidad, ya que el software es la tecnología más importante en el ámbito mundial (Pressman, 2005). A pesar de esto, existen muchos ejemplos de software con mala calidad, lo que resulta en la insatisfacción de los usuarios y en pérdidas económicas (Howles, 2003). Jung et. al (2004) mencionan que el rápido crecimiento de las TIs y PCs resulta en un gran número de software, pero que frecuentemente los usuarios se encuentran insatisfechos con su calidad. Adicionalmente, explican que la satisfacción de los usuarios es considerada frecuentemente como un resultado crítico de la administración de la calidad, describen que estudios previos muestran que tiene un impacto positivo en costos, utilidades y crecimiento de ventas. Evaluar la calidad en el software es una tarea complicada (López De Luise & Agüero, 2007).Generalmente los proyectos de desarrollo de sistemas de información (SI) implementan planes de calidad durante el proyecto, esto proporciona una manera ordenada y

controlada para llevar a cabo el proceso de desarrollo. Pero esto no garantiza que el SI tendrá calidad. Schneidewind (2002) argumenta que la medición de la calidad es clave para el desarrollo de productos de alta calidad. Sobre todo porque la evaluación depende, en gran parte, de la visión del usuario final. El éxito de un SI, más que depender de un nivel de calidad evaluado internamente, se relaciona con el grado de aceptación y satisfacción del usuario que toma como base si el SI cumple con sus necesidades funcionales y operativas. Por ejemplo, los usuarios asignan una buena calidad al software si hace lo que ellos desean, en una forma que sea fácil de aprender y fácil de utilizar (Pfleeger, 2008).

Muchos de los riesgos en el desarrollo de SI se deben a que los desarrolladores prueban los productos y no siempre están familiarizados con las áreas de negocio relacionadas al proyecto. Por otro lado, los usuarios son quienes realmente saben cuáles son sus necesidades y los resultados que esperan obtener. Existen aspectos propios del usuario tales como estilo de aprendizaje, herramientas favoritas, e incluso diferencias culturales, los cuales pueden impactar en la eficiencia del sistema en cuestión (Macracken & Wolfe, 2004). Este aspecto humano de la Ingeniería de Software puede ser el más crítico, ya que un SI puede tener un desempeño excelente realizando una función, pero si los usuarios no pueden entender cómo usarlo, el sistema es un fracaso (Pfleeger, 2008). Entonces, es importante que lo usuarios sean integrados de una manera formal dentro del ciclo de desarrollo de software, tanto en la definición de planes de prueba como en la evaluación del producto.

Existen diferentes modelos de calidad que indican cuáles son las cualidades deseables para determinar la calidad de un producto de software y aunque se definen criterios para medir calidad, es necesario establecer métodos de evaluación cuantitativa que permitan obtener resultados objetivos, integrando tanto los elementos técnicos a evaluar como el involucramiento del usuario. ¿Qué es calidad? El ISO (ISO, 2000) define calidad como: "Grado en el que un conjunto de características inherentes cumple con los requisitos" (p. 23). Donde una característica es un rasgo diferenciador y un requisito es una necesidad o expectativa establecida, generalmente implícita u obligatoria. Dolado y Fernández (2000) la definen como la valoración por los consumidores de la superioridad o excelencia de un bien o servicio. Garvin (1988) distingue tres clases de calidad: *subjetiva*, relacionada con los atributos de bienes y servicios, así como el valor asignado por el cliente; *objetiva*, alude al desarrollo del bien o servicio y a su grado de cumplimiento respecto a las especificaciones establecidas y *rentable*, evalúa si las características de diseño (calidad objetiva) son adecuadas a las necesidades del cliente y si ambas condiciones se cumplen de manera eficiente.

En la Ingeniería de Software, la CS es la concordancia con los requerimientos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo documentados y con las características implícitas que se esperan de todo software desarrollado profesionalmente (Pressman, 2005). Además, esta debe integrarse en todo el ciclo de vida del mismo.

Generalmente el software se desarrolla de manera intuitiva, con la participación de los usuarios en las etapas iniciales del ciclo de vida; lo que generalmente provoca que el desarrollador tienda a interactuar muy poco con ellos. No incluir al usuario puede traer como consecuencia software difícil de utilizar, así como pérdidas económicas y altos costos de mantenimiento. Proyectos de esta forma pueden resultar SI rechazados por los usuarios, o simplemente no se podrían cumplir las metas de la organización para dicho sistema. Macracken y Wolfe (2004) mencionan que algunos problemas que acarrea el no realizar un diseño centrado en el usuario pueden ser la pérdida significativa de tiempo de desarrollo, altos costos en diseño y mantenimiento, y pérdidas económicas para corregir los errores. En consecuencia, se puede decir que la calidad de un producto de software no depende únicamente de seguir fielmente los procesos de desarrollo sino que también impacta el grado de satisfacción del usuario al utilizarlo. De acuerdo a Pfleeger (2008) un sistema puede tener un excelente desempeño, pero si los usuarios no lo utilizan, entonces el sistema es un fracaso total.

En este trabajo se propone una forma de evaluar software, por medio del uso de modelos de calidad. Esta evaluación se centra principalmente, la hace el usuario final, ya que es quién finalmente acepta o rechaza el sistema. Esta propuesta pretende identificar si se puede mejorar el proceso de desarrollo con el uso de modelos para mejorar la satisfacción de los usuarios. Resultados previos en experimentos similares nos mostraron que con muestras pareadas los resultados son la calidad del sistema evaluado se incrementó. Es debido a esto que para la presente investigación el sistema fue evaluado por grupos de usuarios independientes y confirmar las evaluaciones previas.

Gestión de la Calidad.

Para poder implementar la calidad dentro de una organización, se necesita realizar la Gestión de Calidad (GC) y consiste en realizar actividades coordinadas para dirigir y controlar una organización en lo relativo a la calidad (ISO, 2000). Esta se basa en la determinación y aplicación de políticas y se aplica normalmente a nivel empresa, pero también puede haber una a nivel de cada proyecto de software. Los interesados en la evaluación de la CS son la base de muchas decisiones importantes (Jung, et al., 2004) en proyectos de SI. Sommerville (2006) indica que la GC del software se estructura en tres actividades: *Aseguramiento de la Calidad, Planeación de la Calidad y Control de la Calidad.* De manera general, para lograr la CS existen dos componentes imprescindibles que deben intersectarse: calidad del proceso y calidad del producto. Un software que sólo cuide uno de los dos tiene el enorme riesgo de fallar. Pressman (2005) argumenta que la calidad del proceso de desarrollo afecta directamente a la calidad de los productos. Sommerville (2006) indica que el mejoramiento de la calidad se centra en identificar buenos productos de calidad, examinar sus procesos y generalizarlos para aplicarlos posteriormente. Sin embargo, la relación entre el proceso de software y la calidad del producto es compleja. Asimismo, argumenta que la administración de la calidad del proceso comprende: definir estándares de proceso, supervisar el proceso de desarrollo para asegurar que se sigan los estándares y hacer informes del proceso para la administración del proyecto y para el cliente. Cambiar el proceso no siempre conduce a mejorar la calidad del producto. La Figura 1 muestra los cuatro factores que afectan la calidad del producto.

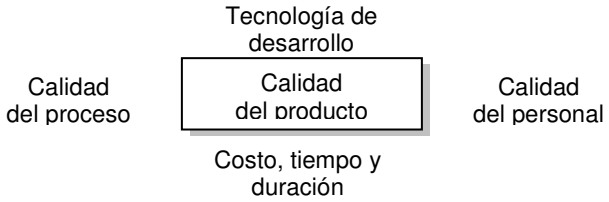

Figura 1. Factores de calidad del producto (adaptado de Sommerville, 2006)

Estándares de calidad.

El aseguramiento de la calidad es la parte de la GC orientada a proporcionar confianza en que se cumplirán los requisitos establecidos (ISO, 2000). Pressman (2005) menciona que el aseguramiento de la calidad del software (ACS) es el conjunto de actividades planificadas y sistemáticas necesarias para aportar la confianza que el software cumplirá los requisitos establecidos. Para lograr el ACS es necesaria la utilización de modelos y estándares establecidos, o bien, establecer un nuevo modelo basado en la combinación de los existentes. Los modelos de calidad integran la mayor parte de las mejores prácticas, proponen temas de administración en los que cada organización debe hacer énfasis, integran diferentes prácticas dirigidas a los procesos clave y permiten medir los avances en calidad (Piattini & García-Rubio, 2003). Por otro lado, los estándares de calidad permiten definir un conjunto de criterios de desarrollo que guían la forma en que se aplica la Ingeniería del Software y suministran los medios para que todos los procesos se realicen de la misma forma y son una guía para lograr la productividad y la calidad (Piattini & García-Rubio, 2003).

La CS es una compleja combinación de factores que variarán entre las diferentes aplicaciones y los clientes que las solicitan (Pressman, 2005) y son: medibles directamente y medibles indirectamente. Es importante que en ambos se compare el software contra algún conjunto de datos y obtener algún indicador de calidad. Una clasificación de los factores que afectan la CS se concentran en tres aspectos, los cuales se muestran en la Figura 2. Todos son muy importantes, imprescindibles y medibles de formas diferentes.

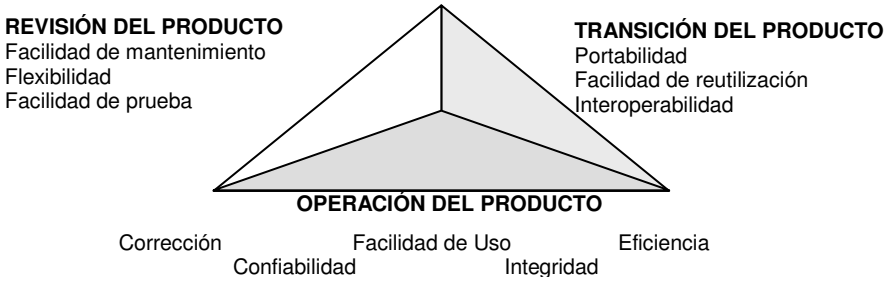

Figura 2. Factores de calidad de McCall (adaptado de Sommerville, 2006)

El ISO/IEC 9126 (ISO/IEC_9126-1, 2001) es un estándar que intenta identificar los atributos de calidad para el software. Este define un modelo formado por 6 características y 27 subcaracterísticas (Jung, et al., 2004) y estos son: *Funcionalidad*: idoneidad, exactitud, interoperabilidad, cumplimiento y seguridad; *Confiabilidad*: madurez, tolerancia a fallas y facilidad de recuperación; *Facilidad de uso*: facilidad de comprensión, facilidad de aprendizaje y operabilidad; *Eficiencia*: comportamiento en el tiempo, comportamiento de los recursos; *Facilidad de mantenimiento*: facilidad de análisis, facilidad de cambio, estabilidad y facilidad de prueba; y *Portabilidad*: adaptabilidad, facilidad para instalarse, cumplimiento, facilidad para reemplazarse. Debido a dudas acerca del papel acerca de los estándares, es importante investigar si el ISO/IEC 9126 es confiable para evaluar la satisfacción de los usuarios acerca de la calidad (Jung, et al., 2004).

El estándar ISO/IEC 90003:2004 describe la Planificación de la Calidad del Software (PCS) como la parte de la Gestión de la Calidad encargada de realizar el proceso administrativo de desarrollar y mantener una relación entre los objetivos y recursos de la organización; y las oportunidades cambiantes del mercado. Esta inicia en las primeras etapas del proceso de software. Un PCS define la calidad del producto deseado, cómo valorar esta calidad y el significado de software de "alta calidad". El plan de calidad define tanto los atributos más importantes del producto a ser desarrollado así como el proceso de evaluación (Sommerville, 2006).

Control de la Calidad.

Según la Norma ISO 9000:2000, el control de la calidad es la parte de la GC orientada al cumplimiento de los requisitos. El Control de la Calidad del Software (CCS) son las técnicas y actividades de carácter operativo utilizadas para satisfacer los requisitos relativos a la calidad, centradas en 2 objetivos fundamentales: mantener bajo control un proceso y eliminar las causas de los defectos en las diferentes fases del ciclo de vida (Pressman, 2005). Sommerville (2006) menciona que existen 2 enfoques complementarios para el CCS:

1. Revisiones de calidad en las que el software, documentación y procesos utilizados para producirlo software son revisados por un grupo de personas.
2. Valoración automática del software en la que el software y documentos producidos se procesan por algún programa y se comparan con estándares aplicables a ese proyecto.

Medición y Métricas de Software

La medición del software consiste en derivar un valor numérico para algún atributo (Sommerville, 2006) y pretende: ayudar a entender qué ocurre durante el desarrollo y el mantenimiento, permitir controlar qué es lo que ocurre en nuestros proyectos y controlar los procesos y productos (Fenton, 1991; Pfleeger, 2008). La medición permite mejorar la calidad, ya que así se identifica el grado actual y compara con alguna norma. Pressman (2005) explica que el proceso de medición se debe realizar mediante cinco actividades: *Formulación*, *Recolección*, *Análisis, Interpretación* y *Retroalimentación*.

Una métrica de software es una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado (IEEE, 1994) y es cualquier tipo de medida relacionada con un sistema, proceso o documentación de software (Sommerville, 2006). Las entidades cuyos atributos se pueden medir en el software, se clasifican en 3 de los cuales se derivan sus respectivas métricas (Dolado & Fernández, 2000):

- *Procesos*. Conjunto de actividades relacionadas con el software que habitualmente incluyen el factor tiempo. Son Las métricas de proceso cuantifican atributos como tiempo, esfuerzo, número de incidencias, etc. Por ejemplo: esfuerzo y tiempo promedio requeridos para reparar los errores reportados.
- *Productos*. Son el resultado (programas, documentación y otras entregas) de la actuación de los procesos. Las métricas del producto incluyen el tamaño, la complejidad de la estructura de datos y el tipo de software.
- *Recursos*. Son el conjunto de elementos que son entradas para la producción del software.

Fenton (1991) argumenta que al medir el software, se suele clasificar en Atributos internos y Atributos externos y los clasifica de acuerdo a la Tabla 2.

Entidad	Atributo interno	Atributo externo
Producto		
Especificaciones	Tamaño, reutilización, modularidad, redundancia, funcionalidad, sintaxis, corrección …	Comprensibilidad, mantenibilidad
Diseño	Tamaño, reutilización, modularidad, cohesión, acoplamiento, funcionalidad …	Calidad, complejidad, mantenibilidad
Código	Tamaño, reutilización, modularidad, acoplamiento, funcionalidad, complejidad algorítmica, …	Confiabilidad, usabilidad, mantenibilidad, reusabilidad
Test	Tamaño, cobertura	Calidad, reusabilidad
Procesos		
Construcción de especificaciones	Tiempo , esfuerzo, nº de cambios	Calidad, costo-efectividad
Diseño	Tiempo, esfuerzo, nº de errores en las especificaciones	Efectividad-Costos
Pruebas	Tiempo, esfuerzo, nº de errores encontrados	Costo, costo-efectividad, estabilidad

Tabla 1. Clasificación de métricas (Adaptada de (Fenton, 1991))

Importancia de la Calidad del Software

Pressman (2005) indica que en el desarrollo de software, la calidad del diseño incluye requisitos, especificaciones y el diseño del sistema. La calidad de concordancia es un tema enfocado principalmente en el diseño de la implementación. Si ésta sigue el diseño y el SI resultante satisface sus requisitos y metas de desempeño, la calidad de concordancia es alta. Pero, ¿la calidad del diseño y la calidad de concordancia son los únicos temas que deben considerar los ingenieros de software? Glass (1998) afirma que la calidad es importante, pero que si el usuario no está satisfecho minimiza dicha importancia. Además, argumenta que es conveniente una relación más intuitiva dando la siguiente fórmula:

Satisfacción_usuario=producto_manejable+buena_calidad+entrega_en_tiempo_y_presupuesto

La CS es una rama muy importante en la Ingeniería de Software debido al acelerado crecimiento que ha tenido la industria, lo cual ha llevado a una búsqueda constante de técnicas de mejoramiento. Esta puede enfocarse tanto al proceso como al producto, por lo que su determinación puede darse como una combinación de ambos enfoques. Dolado y Fernández (2000) argumentan que la preocupación por la CS se ha centrado en asegurarlo a través de los procesos de acuerdo con los estándares tradicionales adoptados en otras ingenierías. Este enfoque puede proporcionar buenos resultados en donde existe un paralelismo evidente entre los criterios de valoración del usuario y los factores técnicos que lo determinan, pero no parece adecuado para aquellas en las que los criterios determinantes de la elección del cliente difieren bastante de los factores técnicos, no son fácilmente identificables o resultan de una combinación aún desconocida de atributos que sólo en conjunto adquieren un significado funcional. Además, argumentan que el empleo de factores en la industria del software como la ingeniería humana, la transición y la explotación del producto o la facilidad de prueba, no parecen ser los criterios de evaluación más apropiados para valorar la calidad desde la perspectiva del usuario, ya que se asume que el alcance de la excelencia, en lo que a calidad se refiere, es la consecuencia de un proceso de desarrollo enfocado a su consecución. Sin embargo, la calidad percibida por el cliente remite a la confirmación de sus expectativas sobre el producto desarrollado, mientras que la calidad técnica por definición, ignora cualquier factor no técnico, siendo que el usuario es quien determina el nivel de excelencia de referencia. Asimismo, sugieren disponer de un instrumento que mida la satisfacción del usuario, lo cual puede ser una buena solución al problema de los productos de mala calidad.

Por otro lado, Amasaki et al. (2005) afirman que un software con calidad ayuda a reducir costos de mantenimiento y satisfacer los requerimientos de los usuarios. Existe un amplio rango de atributos de calidad potenciales a considerar durante el proceso de planeación de la calidad (Sommerville, 2006). Voas (2004) explica que los atributos de software son una colección de comportamientos estrechamente relacionados que por sí mismos tienen poco o ningún valor para los usuarios pero que si se agregan pueden incrementar significativamente el valor de un sistema. El objetivo es identificar los atributos que el usuario desea y reconocer que los atributos pueden ser personalizados de acuerdo a individuos o grupos de usuarios. Entonces, la calidad es algo abstracto que depende la percepción de cada usuario. En particular, la calidad de un software es juzgada de acuerdo a su interfaz de usuario (Pressman, 2005). Como en general no es posible optimizar todos estos atributos, una parte importante de la planeación será seleccionar los más importantes y planear cómo alcanzarlos. A este respecto, Zayaraz et al. (2005) argumentan que la CS tiene diversos aspectos, pero la calidad general puede expresarse en términos de la combinación de varios de ellos.

Entonces, consideramos que debe definirse qué atributos pueden determinar la CS así cómo su evaluación por parte de los usuarios.

Ramani (2007) propone dos técnicas de evaluación del software desde el punto de vista de calidad en el uso: la Evaluación basada en cuestionarios y la Evaluación basada en métricas. Adicionalmente, menciona que incrementando la calidad se pueden obtener beneficios adicionales tales como: incremento de la eficiencia, mejora de la productividad, reducción de errores, reducción en la capacitación y mejora de la aceptación. Otros (Amasaki, et al., 2005; Zayaraz, et al., 2005) indican que la CS se mejora reduciendo el número de fallas mediante pruebas exhaustivas. De igual forma, Sommerville (2006) explica que se ha tratado de reducir la incertidumbre en el control de calidad mediante el control estadístico de la misma. El proceso se mejora con el propósito de reducir el número de defectos. Además, la mejora será continua hasta que los resultados del proceso sean predecibles y el número de defectos reducidos. Posteriormente, se estandariza e inicia un ciclo de mejora permanente.

Con todo lo anterior, se enfatiza el impacto de la CS lo que ha llevado a una búsqueda constante de técnicas que permitan obtener cada vez mejores productos de software y es claro que uno de los retos más importantes es cómo lograr y evaluar dicha calidad. Consideramos que la liberación de un software sin probar adecuadamente lleva implícito un alto grado de vulnerabilidad. Entonces, es importante contemplar que el método de evaluación del software se integre de manera formal dentro del proyecto de desarrollo, buscando que el producto que se entregue al usuario cuente con una calidad óptima.

Ramani (2007) se enfoca a la medición de la satisfacción del cliente en base a la calidad de uso, enfatizando que los productos deben estar creados con una técnica excelente y ser fáciles de usar. Hace referencia a los conceptos de calidad establecidos en el estándar ISO/IEC 8402. De la misma manera, él referencia el modelo de calidad ISO-9126. Adicionalmente, enfatiza el uso de métricas de calidad.

Con el análisis de las investigaciones, es notable el énfasis que se hace en la calidad durante el proceso de desarrollo, centrándose principalmente en la arquitectura del software y únicamente Ramani (2007) se enfoca hacia el concepto de la calidad orientada al uso y satisfacción del usuario, que es justamente la línea de la presente investigación.

Objetivo General

Aplicar un método para la evaluación de un producto de software donde los usuarios finales participen, tomando como base factores de calidad de modelos existentes, evaluar de manera iterada y con la utilización de técnicas estadísticas para obtener un producto con calidad hasta lograr un nivel ideal por los usuarios. Por lo tanto, se plantean las siguientes hipótesis:

H1. La evaluación de la Funcionalidad de un producto de software por parte del usuario final mejora significativamente su calidad.

> **H1a.** La evaluación de la Conformidad de un producto de software por parte del usuario final mejora significativamente su calidad
>
> **H1b.** La evaluación la Exactitud de un producto de software por parte del usuario final, mejora significativamente su calidad
>
> **H1c.** La evaluación de la Seguridad de un producto de software por parte del usuario final mejora significativamente su calidad

H2. La evaluación de la Confiabilidad de un producto de software por parte del usuario final mejora significativamente su calidad

> **H2a.** La evaluación de la Madurez de un producto de software por parte del usuario final mejora significativamente su calidad
>
> **H2b.** La evaluación de la Tolerancia a Fallas de un producto de software por parte del usuario final mejora significativamente su calidad
>
> **H2c.** La evaluación de la Recuperabilidad de un producto de software por parte del usuario final mejora significativamente su calidad

H3. La evaluación de la Facilidad de Uso de un producto de software por parte del usuario final mejora significativamente su calidad.

H3a. La evaluación de la Facilidad de Compresión (Comprensibilidad) de un producto de software por parte del usuario final mejora significativamente su calidad

H3b. La evaluación de la Facilidad de Aprendizaje de un producto de software por parte del usuario final mejora significativamente su calidad

H3c. La evaluación de la Operabilidad de un producto de software por parte del usuario final mejora significativamente su calidad

H4. La evaluación del Comportamiento del tiempo de respuesta de un producto de software por parte del usuario final mejora significativamente su calidad

H5. La evaluación de la Adaptabilidad de un producto de software por parte del usuario final mejora significativamente su calidad

DESARROLLO DE LA INVESTIGACIÓN

Para la presente investigación se desarrolló un método que consistió en establecer, en base a los modelos y estándares de calidad analizados (McCall e ISO 9126), qué factores de calidad de un producto de software deben evaluarse desde el punto de vista del usuario, cómo deben medirse y en qué términos. Una vez obtenidos los criterios definidos, se diseñó un instrumento con el cual se evaluó una aplicación de software en dos iteraciones por dos grupos de usuarios diferentes. La selección de estos modelos se basa en el hacho de que otras alternativas (CMMI, MoProSoft, etc) se centran en la evaluación de la calidad del proceso más no del producto. La literatura reporta que la calidad en el proceso no garantiza la calidad en el producto, solo estandariza la forma de realizar las actividades.

Para la evaluación de los factores a analizar se utilizó estadística descriptiva y se obtuvo un resultado medible y comparable de forma tal que se pueda apreciar de manera concreta la calidad asignada por el usuario al software evaluado. En la Figura 3 se muestra el proceso de desarrollo de software base para la realización del método desarrollado, dentro del cual se enfatiza el paso *Pruebas de calidad (usuarios)* con el objetivo de que al ser parte integral del proceso de desarrollo para garantizar de forma inherente la satisfacción del usuario.

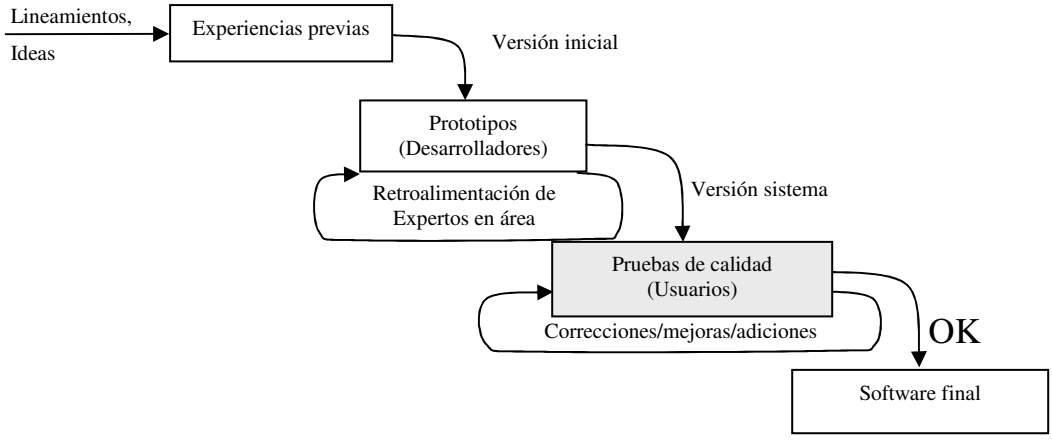

Figura 3. Modelo de desarrollo (Gómez-Reynoso & Brizuela-Sandoval, 2008)

Para la presente investigación se analizaron los modelos de calidad de McCall e ISO/IEC 9126 mediante el cruce e integración de sus atributos, identificando cuáles de ellos podían evaluarse desde el punto de vista del usuario final (Ver Figura 4 y Tabla 2). Ambos modelos fueron seleccionados ya que concuerdan en muchos de los factores evaluables. Existen otros factores en los modelos integrados pero son evaluables desde el punto de vista del desarrollador, lo cual queda fuera de la presente investigación.

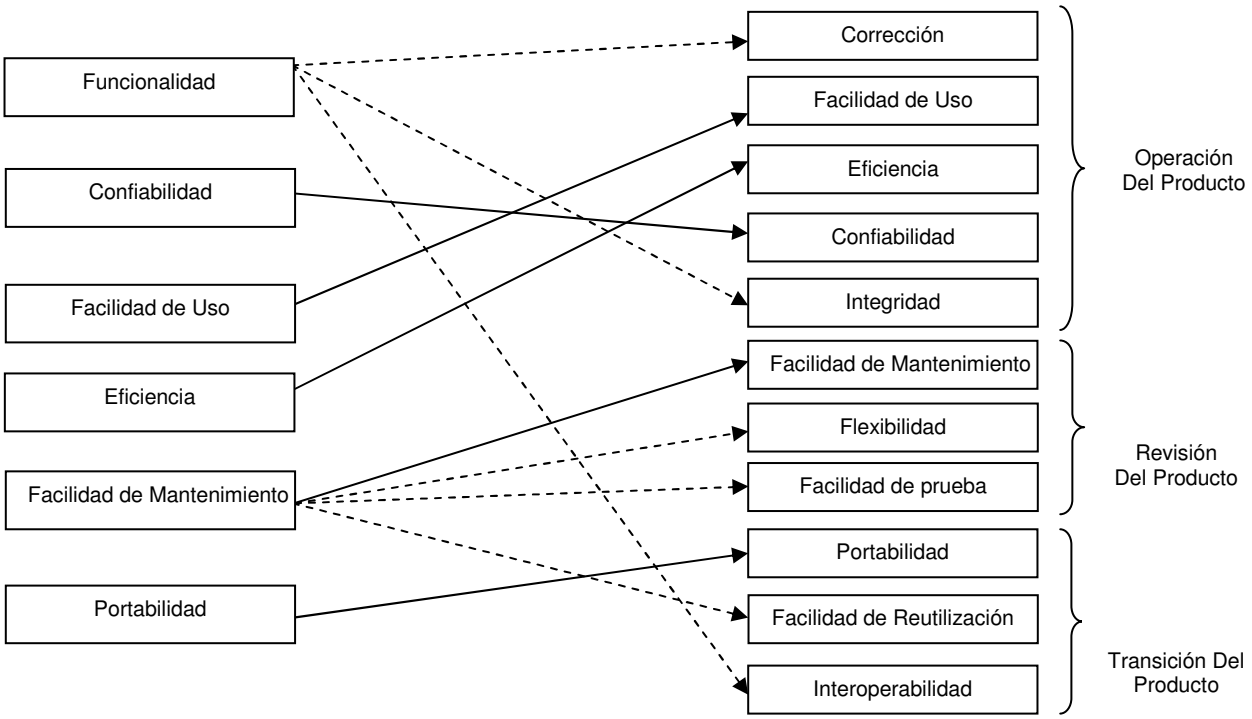

Figura 4. Equivalencias McCall - ISO 9126

Característica de Calidad	Factor	Descripción
Funcionalidad	Conformidad	El software tiene capacidad de proporcionar funciones adecuadas
	Exactitud	El software proporciona resultados correctos, precisos y satisface los objetivos de cliente
	Seguridad	El software cuenta con mecanismos de seguridad y control de accesos no autorizado a los datos
Confiabilidad	Madurez	El software tiene la capacidad de evitar errores
	Tolerancia a Fallas	El software está preparado para mantener un desempeño adecuado en caso de fallas o errores
	Recuperabilidad	El software que considera las acciones necesarias para recuperarse en caso de fallas de manera fácil y rápida
Facilidad de Uso	Comprensibilidad	El software es fácil de entender en su conveniencia y en cómo puede ser utilizado
	Facilidad de Aprendizaje	El software permite al usuario aprender su utilización de manera fácil y rápida
	Operabilidad	El software facilita al usuario su operación y control
Eficiencia	Tiempo de Respuesta	El software cuenta con tiempos adecuados de respuesta al realizar sus diferentes funciones
Portabilidad	Adaptabilidad	El software puede adaptarse a diferentes entornos sin la necesidad de aplicar acciones específicas diferentes a las establecida

Tabla 2. Factores para evaluar la calidad de software

Estudio Piloto

En base a la revisión de la literatura y las teorías base sobre CS, modelos y estándares de calidad analizados (McCall e ISO 9126), se identificaron los elementos de calidad de un producto de software que pueden ser percibidos por los usuarios.

A partir de los elementos identificados, se crearon las preguntas así como la escala de respuestas posibles. Con esto, se integró un cuestionario inicial, y se aplicó en un estudio piloto a 14 alumnos de la asignatura de Ingeniería de Software en el 9º semestre de Ingeniería en Sistemas Computacionales. Para tal efecto, los estudiantes evaluaron con el instrumento creado, la calidad de una SI que se les entregó.

La aplicación del estudio piloto permitió la retroalimentación de los alumnos, en cuanto a la utilidad del instrumento. De esta forma, se identificaron y corrigieron problemas potenciales en el diseño, dando como resultado el instrumento final de evaluación de calidad de los productos de software que quedó conformado por 25 preguntas. Todas las preguntas tienen una escala de 1 (Calidad Excelente) a 7 (Pésima Calidad).

Estudio Final

En el estudio final participaron como usuarios empleados que utilizan aplicaciones de software como parte de su actividad diaria. Para la recolección de datos, se usó como caso de estudio una aplicación dentro del Sistema Integral de Información y Modernización Administrativa en una universidad pública. El sistema integra el manejo de: Control Escolar, Finanzas y Recursos Humanos. Esta fue elegida ya que es utilizada cotidianamente por usuarios de diferentes perfiles. Esta aplicación está integrada por 5 apartados, los cuales se describen brevemente cada una:

- Peticiones. Contiene el listado de peticiones de acuerdo a filtros seleccionados y es la opción principal.
- Detalles/Captura. Realiza la administración general de peticiones, los comentarios relacionados y, en su caso, la calificación otorgada a su atención por medio del usuario.
- Objetivos. Permite realizar las operaciones relacionadas al manejo de objetivos de una petición, tales como asignación, modificación, seguimiento.
- Reporte Semanal. Permite generar reportes de peticiones y objetivos asignados a un empleado para su atención.
- Consulta de Objetivos. Permite una consulta global de los objetivos registrados en las diferentes peticiones.

La aplicación del instrumento se realizó por invitación enviando el cuestionario correspondiente vía correo electrónico a los usuarios seleccionados. La primera iteración se realizó en base a la versión existente de la aplicación que se encontraba de manera oficial dentro del sistema y se obtuvieron un total de 29 cuestionarios contestados. La segunda iteración se realizó en base a la versión corregida que se hizo de la versión original en base a los resultados de la primera. En esta etapa participaron 26 usuarios diferentes a los de la primera. A continuación se describen los datos demográficos de los participantes.

Primera Iteración

Las porcentajes por edades son como sigue: 21-25 y 36-40 años, 31%; 41 ó más 20.7% y 26-30 (3.4%). En cuanto a género: masculino (69%) y 31% femenino. Respecto al nivel de estudios, licenciatura, 51.7%; maestría, 24.1%; secundaria-bachillerato, 14.2% y especialidad, 6.9%; esto puede provocar que los participantes tiendan a ser más críticos de la aplicación y demandar una mejoría notable de la aplicación, lo cual es muy bueno para la presente investigación. En cuanto al Tipo de Puesto, empleados de confianza, 58.6%; sindicalizados, 37.6% y gerenciales, 3.4%. Finalmente, en relación con los antecedentes académicos, Sistemas de Información, 75.9%, Administración, 17.2% y Otros, 6.9%. Consideramos que el contar con usuarios de diferentes áreas ayuda a obtener mejores resultados.

Segunda Iteración

Las porcentajes por edades son: 21-25 y 36-40 años, 30.8%; 41 ó más 26.9% y 31-35, 7.7% y 26-30, 3.8%. En cuanto a género: masculino (65.4%) y 34.6% femenino. Respecto al nivel de estudios, licenciatura, 57.7%; maestría, 23.1% y secundaria-bachillerato, 19.2%. En cuanto al Tipo de Puesto, empleados de confianza, 61.5% y sindicalizados, 38.5%. Finalmente, en relación con los antecedentes académicos, Sistemas de Información, 80.8%, Administración, 15.4% y Otros, 3.8%. En general, los datos muestran aproximadamente la misma distribución a la primera iteración, por lo que se descarta sesgo.

ANÁLISIS DE DATOS

Con el fin de evaluar la calidad de cada aspecto y tomar decisiones al respecto se consideró que un valor promedio entre 1 y 1.5 como *Excelente*, 1.5 a 2 como *Muy Buena*; 2 a 2.5 es *Mínima Aceptable*; cualquier valor superior a 2.5 fue considerado como calidad *Mediocre*.

Primera Iteración

La Tabla 3 muestra los resultados en la primera iteración respecto a la calidad inicial del SI evaluado.

Estadísticas	Confor-midad	Exactitud	Segu-ridad	Madurez	Tolerancia a fallas	Recupera-bilidad	Compren-sibilidad	Facilidad de aprendizaje	Opera-bilidad	Tiempo de respuesta	Adapta-bilidad
Media	2.2276	2.2069	1.8793	3.1724	2.6552	3.2759	2.2241	2.4713	2.6034	2.6552	3.2759
Mediana	2.0000	2.0000	1.7500	3.0000	2.0000	3.0000	2.0000	2.3333	2.5000	2.0000	3.5000
Moda	2.00	2.00	1.00[a]	2.00	2.00	2.00	2.00	1.67	2.50	2.00	3.50
Desviación Estándar	.64081	.90156	.79513	1.53690	1.20344	1.81061	.77444	.92375	.90019	1.14255	1.08015

a. Existen múltiples modas. Se muestra el valor más pequeño

Tabla 3. Estadísticas Primera iteración

Sólo *Seguridad* (1.87) indica que los usuarios opinaron que la versión de la aplicación evaluada tenía muy buena calidad, pero se consideró recomendable revisar qué aspectos estaban afectando, para aproximarla a una calidad totalmente satisfactoria. El resto de ellos definitivamente necesitan mejorarse sustancialmente.

Teniendo ya identificados los aspectos con calidad insuficiente, se realizaron modificaciones a la aplicación de software con intención de mejorar la calidad de forma significativa. Una vez hechas las modificaciones se procedió a la segunda evaluación.

Segunda Iteración

La Tabla 4 muestra los resultados obtenidos en la segunda iteración.

Estadísticas	Confor-midad	Exactitud	Segu-ridad	Madurez	Tolerancia a fallas	Recupera-bilidad	Compren-sibilidad	Facilidad de aprendizaje	Opera-bilidad	Tiempo de respuesta	Adapta-bilidad
Media	1.7923	1.6923	1.5577	2.3077	2.1154	2.2692	1.8269	1.8205	1.8269	1.9615	2.4231
Mediana	1.7000	2.0000	1.3750	2.0000	2.0000	2.0000	2.0000	2.0000	2.0000	2.0000	2.2500
Moda	1.00[a]	2.00	1.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00
Desviación Estándar	.65356	.54913	.59711	.73589	.65280	.82741	.64718	.55961	.63154	.77360	.82997

a. Existen múltiples modas. Se muestra el valor más pequeño

Tabla 4. Estadísticas Segunda iteración

Se puede identificar que *Conformidad, Exactitud, Seguridad, Comprensibilidad, Facilidad de Aprendizaje, Operabilidad y Tiempo de Respuesta* tienen al menos Muy Buena; *Madurez, Tolerancia a Fallas, Recuperabilidad y Adaptabilidad* Mínima aceptable. En consecuencia, todos los elementos evaluados están en el rango de pasar las pruebas de calidad.

Con el fin de identificar si existe una diferencia significativa en la calidad de una evaluación a otra se probaron las hipótesis a través de muestras independientes. Se obtuvo que en diez de las características evaluadas tienen mejoras significativas (aceptadas), y solo una no (rechazada). Los resultados se muestran a continuación.

Factor evaluado	Conclusión		Prueba de Levene para Igualdad de Varianzas		Prueba t para Igualdad de Medias						
			F	Sig.	t	df	Sig. (2-colas)	Diferencia Media	Diferencia Error Est.	95% Intervalo de confianza de la Dif.	
										Inferior	Superior
Conformidad	Acepta	Igualdad de varianzas asumida	.000	.987	2.492	53	.016	.43528	.17470	.08487	.78569
		Igualdad de varianzas No asumida			2.489	52.106	.016	.43528	.17490	.08434	.78621
Exactitud	Acepta	Igualdad de varianzas asumida	3.461	.068	2.520	53	.015	.51459	.20420	.10501	.92416
		Igualdad de varianzas No asumida			2.585	46.961	.013	.51459	.19906	.11412	.91506
Seguridad	Rechaza	Igualdad de varianzas asumida	.963	.331	1.680	53	.099	.32162	.19139	-.06227	.70551
		Igualdad de varianzas No asumida			1.707	51.487	.094	.32162	.18845	-.05663	.69986
Madurez	Acepta	Igualdad de varianzas asumida	11.623	.001	2.611	53	.012	.86472	.33115	.20052	1.52892
		Igualdad de varianzas No asumida			2.704	41.138	.010	.86472	.31981	.21892	1.51053
Tolerancia a fallas	Acepta	Igualdad de varianzas asumida	7.386	.009	2.033	53	.047	.53979	.26547	.00732	1.07225
		Igualdad de varianzas No asumida			2.096	44.078	.042	.53979	.25755	.02076	1.05882
Recuperabilidad	Acepta	Igualdad de varianzas asumida	10.484	.002	2.600	53	.012	1.00663	.38716	.23009	1.78317
		Igualdad de varianzas No asumida			2.696	40.125	.010	1.00663	.37333	.25217	1.76109
Compresibilidad	Acepta	Igualdad de varianzas asumida	.083	.775	2.051	53	.045	.39721	.19371	.00868	.78575
		Igualdad de varianzas No asumida			2.071	52.758	.043	.39721	.19181	.01245	.78198
Facilidad de aprendizaje	Acepta	Igualdad de varianzas asumida	6.465	.014	3.114	53	.003	.65075	.20895	.23166	1.06985
		Igualdad de varianzas No asumida			3.196	46.827	.002	.65075	.20364	.24104	1.06046
Operabilidad	Acepta	Igualdad de varianzas asumida	1.336	.253	3.663	53	.001	.77653	.21202	.35127	1.20178
		Igualdad de varianzas No asumida			3.732	50.228	.000	.77653	.20805	.35870	1.19435
Tiempo de respuesta	Acepta	Igualdad de varianzas asumida	3.163	.081	2.605	53	.012	.69363	.26627	.15957	1.22770
		Igualdad de varianzas No asumida			2.659	49.469	.011	.69363	.26083	.16960	1.21767
Adaptabilidad	Acepta	Igualdad de varianzas asumida	1.330	.254	3.254	53	.002	.85279	.26204	.32721	1.37836
		Igualdad de varianzas No asumida			3.301	51.841	.002	.85279	.25831	.33440	1.37117

Tabla 5. Pruebas muestras independientes

CONCLUSIONES

De acuerdo a los resultados encontrados, se pudo comprobar que al integrar al usuario en la evaluación de software, ayuda a incrementar la calidad significativamente, en 10 de los 11 factores analizados. Esto puede indicar que los usuarios ponen especial énfasis en ciertos aspectos de los SI que en ocasiones los desarrolladores tienden a omitir. Además, ya que los resultados obtenidos son a partir de dos grupos de usuarios diferentes, podemos decir que este mecanismo se podría utilizar aunque con modificaciones que no pongan demasiada presión ni incrementen los costos de desarrollo. Se pude proceder utilizando una estrategia en donde participe una muestra pequeña y representativa de usuarios.

Lo que es innegable es el hecho que la calidad en un producto que es totalmente abstracto (software) ya que no se puede pesar, medir, sentir su textura, oler, etc. Demanda de la aplicación de evaluaciones que eliminen en lo posible la subjetividad y sean lo más uniforme posibles. Consideramos que los evaluadores ideales son los usuarios finales ya que ellos se verán motivados o no a usar el nuevo SI. En consecuencia, determinan el éxito o fracaso del software.

Finalmente, consideramos relevante la participación de los usuarios en la evaluación de nuevos SI ya que esto puede ayudar a reducir los temores al cambio, bajar las barreras que ponen los usuarios a nuevos sistemas y formas de trabajo. En consecuencia, las organizaciones podrían maximizar el retorno de la inversión que hacen en software.

Limitaciones

Una limitante para la segunda versión fue que la codificación de la aplicación no se encuentra concentrada en su lógica de negocio dentro de la base de datos sino en el lenguaje de programación utilizado para la interfaz, lo cual impedía optimizar tiempos de respuesta.

Una limitante no esperada, fue el hecho de que las áreas administrativas participantes tienen políticas de restricción en sus computadoras.

Finalmente, una limitante general fue la carga de trabajo de los usuarios involucrados, ya que esto hizo lenta la recopilación de datos y no permitió obtener un mayor número de datos en relación a la lista de usuarios planeada inicialmente.

BIBLIOGRAFIA

1. Amasaki, S., Yoshitomi, T., Mizuno, O., Takagi, Y., & Kikuno, T. (2005). A New Challenge for Applying Time Series Metrics Data to Software Quality Estimation. *Software Quality Journal, 13*(2), 177-193.
2. Dolado, J. J., & Fernández, L. (2000). *Medición para la gestión en la Ingeniería del Software*. Madrid, España: Ra-Ma.
3. Fenton, E. N. (1991). *Software Metrics: A rigorous approach*. London, UK: Chapman & Hal.
4. Garvin, D. A. (1988). *Managing Quality. The strategic and competitive edge*. New York: Free Press.
5. Glass, R. (1998). Defining Quality Intuitively. *IEEE Software, 15*(3), 103-104,107.
6. Gómez-Reynoso, J., M., & Brizuela-Sandoval, M. R. (2008). *Participación de los Usuarios en la Evaluación de Sistemas de Información*. Paper presented at AMCIS 2008
7. Howles, T. (2003). Widespread Effects of Defects. *Quality progress, 36*(8), 58-61.
8. Software Engineering Standards Collection, Standard Glossary of Software Engineering Terminology, Std 610 C.F.R. (1994).
9. ISO 9000:2000. Sistemas de gestión de la calidad - Conceptos y vocabulario (2000).
10. Information Technology - Software product evaluation - Quality characteristics and guidelines for their use (2001).
11. Jung, H.-W., Kim, S.-G., & Chung, C.-S. (2004). Measuring Software Product Quality: A Survey of ISO/IEC 9126 quality. *IEEE Software*, 88-92.
12. López De Luise, M. D., & Agüero, M. J. (2007). Aplicación de Métricas Categóricas en Sistemas con Lógica Difusa. *IEsatin America Transaction, 5*(1), 55-61.
13. Macracken, D., & Wolfe, R. J. (2004). *User-Centered Website Development*: Prentice Hall.
14. McCall, J., Richards, P., & Walters, G. (1977). *Factors in Software Quality. Volumes I, II and III*: RADC Reports.
15. Pfleeger, S. L. (2008). *Software Engineering: Theory and Practice. 3rd Edn*. Singapore: Pearson-Education.
16. Piattini, M., & García-Rubio, F. (2003). *Calidad en el desarrollo y mantenimiento del software*. Madrid Ra-Ma.
17. Pressman, R. S. (2005). *Software Engineering: A Practitioner's Approach, 6/e*. New York, N.Y.: Mc Graw-Hill.
18. Ramani, A. (2007). Measuring Client Satisfaction Using Quality in Use. *The quarterly journal of the TickIT software quality certification scheme, 7*(1), 5-11.
19. Schneidewind, N. F. (2002). Body of Knowledge for Software Quality Measurement. *IEEE Computer, 35*(2), 77-83.
20. Sommerville, I. (2006). *Software Engineering. 8/e*. London, U.K.,: Addison-Wesley.
21. Voas, J. (2004). Software's Secret Sauce: The "-ilities". *IEEE Software*, 14-15.
22. Zayaraz, G., Thambidurai, P., Srinivasan, M., & Rodrigues, P. (2005). Software quality assurance through COSMIC FFP. *ACM SIGSOFT Software Engineering Notes, 30*(5), 5 pages.