UNIVERSIDAD JOSÉ CARLOS MARIÁTEGUI

VICERRECTORADO DE INVESTIGACIÓN

FACULTAD DE INGENIERÍA Y ARQUITECTURA

CARRERA PROFESIONAL DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

TRABAJO DE SUFICIENCIA PROFESIONAL

MÉTRICAS DE CALIDAD DE SOFTWARE

PRESENTADO POR:

BACHILLER ERIC JOHAN CÓRDOVA AMPUERO

ASESOR:

MGR. ALBERTO LIMACHE FLORES

PARA OPTAR POR EL TÍTULO PROFESIONAL DE INGENIERO DE SISTEMAS E INFORMÁTICA

MOQUEGUA-PERÚ

2016

CONTENIDO

CAPÍTULO IV

CONCLUSIONES Y RECOMENDASIONES

CONTENIDO DE TABLAS

CONTENIDO DE FIGURAS

RESUMEN

El presente proyecto se desarrolló con el fin de evaluar la calidad de un software aplicando métricas. Para la elaboración del proyecto se hizo una investigación de los métodos de métricas de calidad de software que se ajusten al problema. Teniendo así un consolidado de estos diagnósticos que nos permitió proponer un sistema mediante el cual se hará mención a la medición del software basado en parámetros dados, entre ellos el número de líneas de código o el volumen de documentación asociada. Se utilizará el término indicadores del software. Según las peticiones de los usuarios finales y el aumento de exigencias frente a los productos de software, es preciso utilizar métricas de calidad de software en la fase de desarrollo entonces, es indispensable nuevos métodos de evaluación que dejen saber las apreciaciones de los usuarios.

Palabras clave: calidad, métricas, indicadores, medición.

ABSTRACT

The present project was developed according to set the quality of software by applying metrics. For the development of the project an investigation of the methods of software quality metrics that adjust to the problem was carried out. Having thus a consolidation of these diagnoses that allowed us to propose a system by means of which reference will be made to the measurement of the software based on predetermined parameters, such as the number of lines of code it contains or the volume of associated documentation. The term indicators of the software will be used. Given the claims of the end users and the increase of a largest number of needs compared to software products, it is necessary to use software quality metrics in the development process, therefore, it is crucial to have new evaluation approaches that allow to know the appreciations of the users.

Keywords: quality, metrics, indicators, measurement.

CAPÍTULO I

INTRODUCCIÓN

Los establecimientos hoy en día poseen un elemento de software y hardware, motivo que ha hecho que en el tiempo moderno el progreso de estos, ha creado considerables expectaciones a nivel global de cara al producto final y a la calidad del proceso. De acuerdo a investigaciones presentes, los productos de software han de ser implementados cada vez en menos tiempo y con mayor cantidad de exigencias de usuario, promoviendo de esta forma a la ingeniería de software a elaborar nuevos métodos, procesos y herramientas para la creación de sistemas que tengan en su elaboración particularidades de estabilidad, amigabilidad, seguridad, confiabilidad, eficiencia, parametrizabilidad y lo más importante calidad tanto en el proceso de desarrollo como del producto terminado, según lo definen Akingbehin (2009), Heck (2010), Hofman (2009). De esta manera se dan ciclos de vida puntuales, en el cual la calidad se muestra como un proceso de mejora continua (Pressman, 2008; Calero, 2008; Gall, 2008).

Basado en lo antepuesto es preciso tener para los ingenieros una guía cuantitativa que de una manera de examinar el diseño y la elaboración del software, dándose un conjunto de indicadores y medidas (métricas), las que consientan medir con más exactitud los distintos requerimientos del usuario y del sistema. Así como estas acciones apremian un régimen estándar, hay información y datos que no son fáciles de medir, donde el proceso es subjetivista y la indecisión del usuario final es una variable genérica, por lo tanto, trasforma la idea de calidad, revelando una situación indistinta proporcionada por el eco generado por esta clase de miramientos.

CAPÍTULO II

OBJETIVOS

2.1 Objetivo general

Elaborar un proyecto de software que logre evaluar la calidad del software utilizando métricas orientadas al tamaño.

2.2 Objetivos específicos

Elegir las métricas o medidas apropiadas para medir la calidad del software.

Elaborar una aplicación que pueda evaluar y medir métricas del producto de software.

CAPÍTULO III

DESARROLLO DEL TEMA

3.1 Marco teórico

3.1.1 Software.

El software es el equipamiento lógico e intangible de un computador. La noción de software comprende a todas las aplicaciones informáticas, tales como los editores de imágenes, los procesadores de textos y las planillas de cálculo (Pérez, 2008, p.2).

Mediante diferentes tipos de lenguajes de programación es que se desarrolla el software, que generalmente influyen en la conducta de la máquina. Estos lenguajes son conjuntos de símbolos y reglas semánticas y sintácticas que definen el significado de sus elementos y expresiones. De esta manera a los programadores se les hace sencillo definir que datos deben operar en un computador (Pérez, 2008, p.3).

Entre las tipologías de software, uno de los principales es el software de base o software de sistema, permitiendo al usuario tener el control sobre el hardware (componentes físicos) y dar soporte a otros programas informáticos.

Los llamados sistemas operativos, que empiezan a funcionar cuando se prende el ordenador, son software de base (Pérez, 2008, p.4).

En la actualidad el desarrollo de software es un importante protagonista de la economía mundial, ya que moviliza millones de dólares por año, un claro ejemplo de esto es Microsoft, la cual consiguió propagarse gracias a Windows su sistema operativo y el Office que no es más que una suite de programas de oficina (Pérez, 2008, p.5).

3.1.2 Calidad.

El concepto de calidad puede tener variadas definiciones ya que todo depende del grado de satisfacción del cliente. Por otro lado, se tiene que la calidad es la consecuencia de un esfuerzo complejo, se trabaja de manera eficaz para poder complacer el interés del consumidor. Dependiendo de la manera en que un servicio o producto sea aprobado o no aceptado por los clientes, entonces diremos si es malo o bueno (Olivares, 2013, p.1).

En cuantiosas ocasiones el nivel de calidad se mide por las preferencias y la reacción del cliente. Desde el instante en que el mencionado llega al lugar comercial, sabe precisamente qué va a comprar y dónde encontrarlo, va de frente al sitio donde está el producto de su elección. A veces, no ubicará lo que busca, y por lo tanto optará por otro producto de menor o mayor costo, no obstante, cuando el nivel de predilección se afianza en una marca establecida, el cliente opta por continuar indagando en otros comercios en vez de solucionarse con un producto similar (Olivares, 2013, p.2).

Cuando sucede esto, es muy probable que la calidad de ese producto sea elevada, ya que está consiguiendo que el comprador no lo cambie por otro. La calidad brinda nivel al consumidor, pero no en todos los casos el cliente está dispuesto a invertir en ella. Sin embargo, cuando el consumidor está gastando por un servicio, en muchas ocasiones la calidad de éste será producto de la atención al consumidor y de las mínimas molestias que éste pueda darle (Olivares, 2013, p.3).

Algunas veces, cometemos el error de creer que un servicio o producto es de calidad porque lo oímos o vemos a toda hora en los medios de comunicación. Hay que estar alertas con las publicidades engañosas, y no dejarnos convencer por una marca, sencillamente porque está de moda o vanguardia. El cliente será quien decidirá al final qué es lo que más le conviene (Olivares, 2013, p.4).

3.1.3 Calidad de software.

El desarrollo de la ingeniería de software es crear un sistema, aplicación de calidad o producto, que debe cumplir con las necesidades del usuario para lo cual fue creado. Para alcanzar este objetivo, los ingenieros del software deben emplear herramientas modernas con métodos seguros en un proceso maduro de desarrollo del software (Pressman, 2010, p.582).

Hay un concepto estandarizado de la calidad dado por ISO y dice: es el conjunto de propiedades y características de un software o servicio que le confiere su aptitud para satisfacer las necesidades del usuario final. Es importante indicar que, en la normatividad, asumimos que la calidad de un software debe estar dentro de las distintas fases del proceso de elaboración del producto (International Standard Organization, 2015, p.1).

3.1.4 Evaluación de calidad de software.

Es esencial lograr la medición para cualquier disciplina de la ingeniería y lo podemos apreciar en la ingeniería de software. Lord Kelvin, en una oportunidad, dijo: Cuando pueda medir lo que estás expresando y formularlo con números, ya sabes algo acerca de ello; cuando no logres medir, cuando no logres expresar lo que indicas con números, esta noción es pobre y deficiente: puede ser el inicio del conocimiento, pero en tus pensamientos apenas estas adelantando hacia el escenario de la ciencia (García, 2009, p.2).

La medición logra ayudar al aseguramiento de calidad mediante herramientas habituales, como son: actividades, el control de los procesos y la prueba de sí cumple las exigencias requeridas o si se alcanza un cierto nivel de calidad. A partir de aquí, resulta importante entender la manera que al evaluar la calidad de los productos es mediante la medición, para esto debemos tomar en cuenta que la definición de calidad es compleja como para promediarlo mediante una única medida, quiere decir que, da como resultado inexacto la búsqueda de una valoración que defina con un solo valor o numero la calidad de un producto. Para empezar, el concepto estandarizado de calidad remite a la satisfacción de necesidades formuladas. Por ello, se pretende relacionar la calidad a la satisfacción del usuario (International Standard Organization Tools, 2015, p.1)

De modo general, la calidad es definida como el valor que se le da por el lado de los usuarios de un servicio. A menudo se distinguen tres clases de calidad: objetiva, subjetiva y la rentable. Por calidad externa o subjetiva, es la que se relaciona con las propiedades de los bienes y servicios, así como su valoración por

el cliente. La calidad interna u objetiva, indica el desarrollo del bien o servicio y a su nivel de cumplimiento respecto a las descripciones previamente dadas (Dolado, 2000).

La rentabilidad de la calidad se refiere al efecto que tiene la calidad sobre los costos de la empresa. Para este trabajo, la calidad de referencia es la percibida por el cliente al hacer uso del software (Mora, 2011, p.148).

Por lo tanto, es preciso señalar que al calcular la calidad subjetiva de un producto se efectúa a través de la medición de atributos determinados concentrados en características específicas llamadas métricas. Dentro del ambiente de la calidad de software nace la definición de métrica, esencial para la evaluación del software, por consiguiente, a continuación, se habla de manera global.

3.1.5 Medida, métricas e indicadores.

A pesar que las disposiciones de medida, medición y métricas se usan con frecuencia sin distinción, es esencial recalcar sus diferencias. Los términos medida y medición se pueden confundir, dentro de los conceptos de la ingeniería del software; una medida muestra un indicador cuantitativo de las dimensiones, tamaño y capacidad de los atributos de un producto o proceso. La medición es el acto de establecer una medida. El IEEE (Institute of Electrical and Electronic Engineers) precisa la métrica como una medida cuantitativa del grado en que un componente, sistema o proceso tiene un atributo definido (Pressman, 2010, p.527).

De momento se ha seleccionado un solo factor de los datos (p. Ej. la cantidad de errores sin contener la revisión de un módulo), se ha determinado una medida.

La medición resulta de la recolección de uno o varios factores de los datos (por ejemplo, se indaga una cantidad de revisiones de módulos para reunir medidas de los errores hallados durante cada revisión).

Una métrica del software presenta las dimensiones independientes de algún factor (por ejemplo, el número medio de errores hallados por persona o por revisión y hora de las mismas). Un ingeniero de software reúne medidas y elabora métricas, de tal forma conseguir indicadores.

Un indicador es una métrica o una mezcla de métricas, que proveen un enfoque profundo de los procedimientos del producto o proyecto en sí, además provee una visión profunda que permite a los ingenieros del software o al gestor de proyectos adaptar el proceso, el proyecto o el producto para que todo surja de la mejor manera. Por ejemplo, cuatro grupos de software están elaborando un gran proyecto de software. Cada grupo debe llevar revisiones del diseño, pero puede elegir el tipo de prueba a ejecutar. En cuanto a la evaluación de la métrica, errores hallados por persona y hora gastada, al gestor del proyecto se le informa que dos grupos manejan métodos de revisión formales, muestran errores hallados por persona y hora gastada y es un 40 % mayor que otros grupos. Presumiendo que todos los factores de evaluación son iguales, esto facilita al gestor del proyecto un indicador, en el que los métodos de revisión formales pueden facilitar un ahorro mayor en inversión de tiempo que otras revisiones.

Lo cual propondrá que todos los grupos manejen el enfoque más formal. La métrica facilita al gestor un enfoque profundo y también le lleva a tomar medidas fundadas.

3.1.6 Métrica.

Por término general para la estimación de la calidad, es más usual enfocarse la medición del producto que en medidas de proceso.

En informática, la palabra métrica hace alusión a la medición del software en base a parámetros establecidos, como puede ser el volumen de documentación relacionada o el número de líneas de código que tiene el software. Algunas veces en vez de métrica se usa la expresión indicadores del software. Varios ingenieros lo emplean como sinónimos asimismo otros les cargan conceptos diferentes (Krall, 2006, p.4).

3.1.7 Modelos de evaluación de la calidad de software.

La complejidad de la definición de calidad del software ha ocasionado la exploración de modelos de evaluación de calidad que intentan contribuir con una forma, para precisar esta definición en diversas características sencillas y asequibles de medir o evaluar. Así, se hallan modelos como el modelo de McCall.

Para esto, a continuación, se explica de manera general las características importantes de este modelo, con el objetivo de conocerlo y después, sacar de ellos las métricas necesarias para emplearlas en el software.

3.1.8 Modelo del ciclo de vida de la calidad.

La calidad del producto software consigue ser valorada calculando atributos internos (medidas en su mayoría fijas de productos intermedios), o calculando atributos externos (evaluando la conducta del código mientras se encuentra en ejecución), o evaluando los atributos de aplicación de calidad en uso.

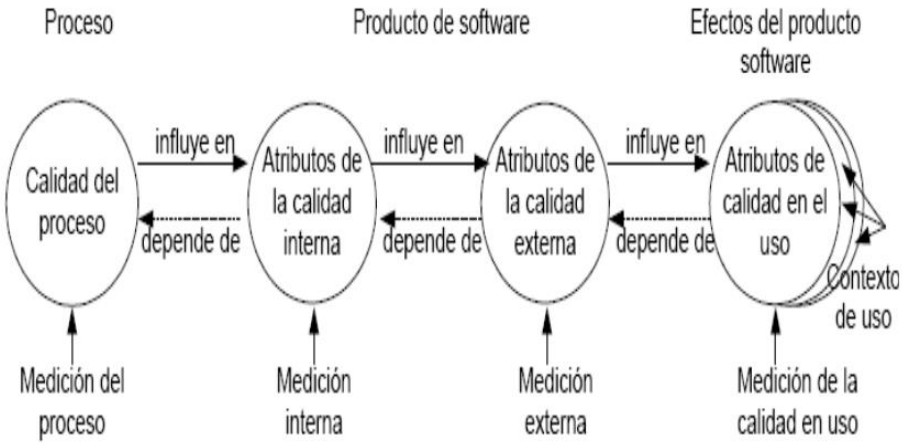

Figura 1. Ciclo de vida de calidad

Fuente: Cochea, 2009

3.1.9 Calidad del producto software y el ciclo de vida.

El usuario exige calidad de forma detallada en requerimientos de calidad por métricas de calidad en uso. La calidad de uso es el punto de vista del usuario desde un contexto y espacio determinado, esta calcula la extensión en la cual los usuarios logran obtener sus metas en un sitio específico.

Cuando se requiere la calidad externa, son dados los requerimientos del usuario, la calidad externa se ejecuta cuando el software esta en operación. La calidad externa es cuando el software es ejecutado, y esta es evaluada y medida en un espacio de simulación.

Los requerimientos de calidad interna son empleados para detallar propiedades.

Las métricas del software es un término que se establece a un extenso rango de varias actividades, por ejemplo:

- Medidas y modelos de estimación de coste y esfuerzo.
- Medidas y modelos de productividad.
- Modelos y medias de calidad.
- Control y Aseguramiento de calidad.
- Recoger datos.
- Modelos de fiabilidad.
- Modelos de evaluación de ejecución.
- Complejidad algorítmica o computacional.
- Métricas de complejidad o estructurales.

Existen en software tres clases de entidades los cuales permiten poder medir sus atributos.

- Procesos: Son acciones de software que usualmente soportan el elemento tiempo. Atributos internos a medir son el esfuerzo, tiempo, número de incidentes de un tipo concreto que se dan en el proceso.
- Productos: Artefactos o documentos generados en el ciclo de vida del software, son entregables. Atributos externos a medir son el entendimiento de un documento de especificación, la mantenibilidad del código fuente, la confiabilidad del código, etc. Atributos internos a medir son la longitud, la funcionalidad, corrección sintáctica de los documentos de especificación o modularidad, etc.

- Recursos: Son todos los elementos que hacen de entrada a la producción software. Los atributos a medir son herramientas, los materiales y métodos, el personal, la productividad del recurso humano, el costo.

3.1.10 Control de la calidad del software.

El control de la calidad se consigue a través de las inspecciones realizadas repetidamente al manejo de herramientas, metodologías de trabajo, verificación de prototipos y de las pruebas formales de los productos finales. Para controlar la calidad, los niveles directivos han de implantar y monitorear conjunto de métricas, que les faciliten información necesaria y así proceder basados en hechos.

3.1.11 Métricas de calidad de sistemas de información.

Las métricas son niveles de unidades sobre las cuales puede medirse un atributo calculable. En software se ha de reunir y analizar datos apoyándose en mediciones reales de software, así como a los niveles de medición. Los atributos son características visibles del proceso o del producto o de software. La expresión producto se emplea para referirse a las especificaciones, diseños y a las listas del código.

Las métricas dan a conocer una medida de que forma el software se ajusta a los requisitos explícitos e implícitos del cliente. Quiere decir, de qué forma debo medir para que mi sistema se adecue a las necesidades que me solicita el cliente.

Las métricas de calidad de sistemas de información se emplean para controlar y examinar el proceso de desarrollo del software, de manera que hacen posible:

- Mostrar la calidad del producto.

- Evaluar la productividad de los desarrolladores.
- Evaluar los beneficios en términos de productividad y calidad.
- Implantar una línea base para la estimación.
- Respaldar el uso de nuevas herramientas o formación agregada.

3.1.12 Las métricas de software.

La medición es esencial para todas las disciplinas de ingeniería, y la ingeniería del software es una de ellas.

Las métricas del software se aluden a una extensa lista de medidas para el software de ordenador. La medición se puede emplear al proceso de software para intentar perfeccionar sobre una base continua.

Podemos explicar las métricas de software o medidas de software como:

La aplicación continúa de técnicas apoyadas en las medidas de los procesos de desarrollo de software y sus productos, para originar una gestión de la información importante y a tiempo. Esta información se empleará para perfeccionar esos procesos y los productos que se adquieren de ellos (Betancourt, 2016, p.1).

3.1.13 Fiabilidad del software.

En la actualidad existen variedad de virus e intrusos informáticos, la integridad del software ha logrado obtener considerable único. Este atributo mide la destreza del producto de software para soportar ataques (tanto accidentales como intencionales) contra su seguridad.

El ataque se puede cometer en cualquiera de los componentes del software, datos, programas y documentos.

3.1.14 Clasificación de métricas.

Las métricas del software se pueden clasificar en medidas directas e indirectas.

- Directas: Contienen el costo y esfuerzo aplicados, las líneas de código (LDC) originadas, velocidad de ejecución (LDC), el tamaño de memoria, y las fallas vistas en determinado tiempo.
- Indirectas: Se refieren a la calidad, funcionalidad, complejidad, fiabilidad, eficiencia, facilidad de mantenimiento, etc.

3.1.15 Métricas de procesos.

Evalúan el proceso en sí de elaboración del producto dado. Ejemplos de este tipo de métricas son el tiempo de elaboración del producto, el esfuerzo que implica dicha elaboración, el número y tipo de recursos aprovechados (personas, máquinas, etc.) el costo del proceso. La oportunidad de obtener este tipo de métricas está relacionada generalmente a alguna técnica de estimación. En el tema siguiente describiremos las técnicas básicas de estimación, y la metodología que se podrá aplicar.

3.1.16 Técnicas de estimación.

Luego de definir las métricas de software se pueden comprender las distintas técnicas existentes para la estimación, la estimación consta especialmente de cuatro técnicas.

3.1.16.1 La opinión de los expertos.

Es la cual está basada en la experiencia profesional de los colaboradores en el proyecto de estimación.

3.1.16.2 La analogía.

Es el historial de proyectos anteriores comparados con el que se va a evaluar. La estimación es ajustable viendo las diferencias entre el proyecto pasado y el nuevo. También se puede decir que es una aproximación más seria que la experiencia de los expertos.

3.1.16.3 La descomposición.

El producto se descompone en componentes más chicos, o un proyecto se descompone en actividades de nivel menor. La estimación se realiza partir del esfuerzo solicitado para generar los componentes más chicos o para hacer las actividades de nivel inferior. Luego de sumar las estimaciones de los componentes de un proyecto resultara la estimación global.

3.1.16.4 Las ecuaciones de estimación.

Son fórmulas matemáticas constituyen el esfuerzo que se tomará y que crean la relación de cualesquiera medidas de entrada (que regularmente es la medida del tamaño del producto).

3.1.17 Lenguajes de programación.

Con la premisa de tener conocimiento de que es un lenguaje cualquiera, lo cual es un sistema estructurado de comunicación, por ejemplo, el lenguaje humano que

hace posible comunicarnos unos con otros por intermedio de signos (palabras, sonidos, gestos, etc.). Comprendiendo lo antepuesto se puede decir que se sabe que es un lenguaje de programación.

Un lenguaje de programación es un sistema estructurado y diseñado esencialmente para que los ordenadores y maquinas se entiendan entre ellos y con nosotros, los humanos contendrán un grupo de acciones consecutivas que el ordenador debe ejecutar.

Estos lenguajes de programación usan distintas normas o bases y se emplean para evaluar de que forma se comporta una maquina (ejemplo, un ordenador), asimismo pueden emplearse para elaborar programas informáticos, etc.

El término programación no es mas que un proceso por intermedio del cual se escribe, codifica, diseña, prueba y se depura un código básico para los ordenadores.

La estimación de recursos, costos y planificación necesita una buena información histórica y el coraje de confiar en datos cuantitativos cuando todo lo que existe son datos cualitativos.

Al ciclo de vida del desarrollo aplicamos estas medidas, cuando debemos estimar los costos, al seguimiento y control de la fiabilidad de los productos finales, y a la forma en que los productos cambian a través del tiempo debido a la aplicación de mejoras.

Un ingeniero de software reúne medidas y desarrolla métricas para conseguir indicadores. Las medidas del software y los modelos de medida son apropiadas para estimar y presagiar costos y para medir la producción y la calidad del producto.

La medición es fundamental si se quiere elaborar un software de calidad.

3.1.18 Mediciones del software.

Para precisar notablemente la definición de métrica, seguidamente, se presentan determinados tipos. Las mediciones del mundo físico se logran clasificar de la siguiente forma: medidas directas (por ejemplo, la longitud de un clavo) y medidas indirectas (por ejemplo, la calidad de los clavos producidos y medidos, calculando los productos con defectos). Las métricas del software se pueden incluir en una categoría de manera parecida.

Entre las medidas directas del producto comprenden la velocidad de ejecución producidas, las líneas de código (LDC), tamaño de memoria y los defectos durante un periodo de tiempo dado. Entre las medidas indirectas, Están la funcionalidad, calidad, eficiencia, complejidad, facilidad de mantenimiento, fiabilidad, etc.

Otra clasificación de métricas, tenemos al tamaño y a la función que cumple el software.

3.1.19 Métricas orientadas al tamaño.

Las métricas del software orientadas al tamaño proceden de la estandarización de las medidas de calidad y/o productividad, teniendo en cuenta el tamaño del

software que se haya originado. Si una organización de software conserva registros simples, se puede hacer una tabla de datos orientados al tamaño, como se ve en la Tabla 1, en la cual se registran los proyectos de desarrollo de software de los últimos años y las medidas que corresponden a cada proyecto.

Tabla 1

Métricas orientadas al tamaño

Proyecto	LDC	Esfuerzo	$(000)	pp.doc	Errores	Defectos	Personas
Alfa	11,900	22	159	354	129	31	3
Beta	26,800	60	429	1199	299	77	5
Gamma	19,800	38	298	1010	235	59	6

Fuente: Pressman, 2010

Refiriéndonos a la entrada de la Tabla 1 del proyecto alfa, se desarrollaron 11900 LDC con 22 personas-mes y con un costo de 159,000 dólares. Debe tomarse en consideración que el esfuerzo y el costo registrado en la tabla contienen todas las actividades de ingeniería del software como es el análisis, diseño, codificación y prueba; también, se muestra que se desarrollaron 354 páginas de documentación, se registraron 129 errores antes de que el software fuera entregado al cliente y se hallaron 31 errores después de la entrega dentro del primer año de uso, asimismo, se sabe que el proyecto fue desarrollado por tres personas.

Para desarrollar métricas que se consigan contrastar entre diferentes proyectos, se eligen las líneas de código como valor de normalización. Con los datos expuestos en la Tabla 1, se pueden elaborar un grupo de métricas simples orientadas al tamaño:

- Errores por KLDC (miles de líneas de código).
- Defectos por KLDC.
- Costo por LDC.

Asimismo, se pueden deducir otras métricas interesantes:

- Errores / personas-mes.
- LDC por persona-mes.
- Costo/página de documentación.

Como paradigma tenemos al LDC y como extensión KLDC.

Indica el número de LDC.

- Entre sus ventajas puede señalarse que es fácil de calcular porque todos los procesadores de textos cuentan con contadores de línea.
- Entre las desventajas puede señalarse que dependen del lenguaje de programación, que no se adaptan fácilmente a lenguajes no procedimentales y que cuando se usa en la estimación demanda un nivel de detalle que no es fácil de conseguir (El planificador debe estimar antes de realizar las etapas de análisis y diseño).

Con esta métrica se puede obtener:

Productividad = KLDC/Esfuerzo

Esfuerzo = persona x mes

Calidad = errores / KLDC

Costo = costo_medio / KLDC

Documentación = páginas de documentación / KLDC

Además, puede obtenerse:

- Errores / esfuerzo
- LDC / esfuerzo
- Dólares / páginas de documentación

3.1.20 Métricas orientadas a la función.

Las métricas del software orientadas a la función, emplean como valor de normalización la funcionalidad dada por la aplicación. La funcionalidad no se logra medir directamente, debe derivarse indirectamente a través de medidas directas. Las métricas orientadas a la función fueron presentadas por primera vez por (Albretch79), quien propuso una medida llamada punto de función. Los puntos de función se resultan con una relación empírica de acuerdo a las medidas contables (directas) del dominio de información del software y las estimaciones de la complejidad del software.

Los puntos de función se calculan luego de completar una tabla (véase la tabla 2). Se establecen cinco características de dominios de información, los valores se especifican de la forma siguiente:

3.1.20.1 Dominio de información.

- Número de entradas de usuario. Se refieren al número de las entradas del usuario que logran facilitar diferentes datos orientados a la aplicación.
- Número de salidas del usuario. Se refieren al número de las salidas que se le facilitan al usuario y la información orientada a la aplicación. En este

contenido la salida se refiere a informes, pantallas, mensaje de error, etc. Los elementos de datos específicos dentro de un informe no se cuentan de forma separada.

- Número de peticiones de usuario. Una petición se precisa como una entrada interactiva que provoca la generación de alguna respuesta instantánea en forma de salida interactiva, se enumera cada petición por separado.
- Número de archivos. Se cuenta cada archivo maestro lógico. Un grupo lógico de datos puede ser parte de un archivo independiente o de una gran base de datos.
- Número de interfaces externas. Se enumeran en su totalidad las interfaces legibles por la máquina, por ejemplo, discos que se utilizan para transmitir información a otro sistema o archivos de cintas.

Tabla 2

Cálculo de métricas de punto de función

	Factor de ponderación			
	Cuenta	**Simple**	**Medio**	**Complejo**
Nro. de entradas de usuario	x	3	4	6
Nro. de salidas de usuario	x	4	5	7
Nro. de peticiones de usuario	x	3	4	6
Nro. de archivos	x	7	10	5
Nro. de interfaces externas	x	5	7	10
Total				

Fuente: Pressman 2010

El concepto de los valores del dominio y la manera de calcular los puntos de función es complicada y debido, a que no está dentro del objetivo de este trabajo, únicamente se señala como referencia.

3.1.20.2 Complejidad del software.

Se considera un único parámetro:

Valores de ajuste: Se refiere a un cuestionario de 14 factores de valoración sobre aspectos de funcionalidad y complejidad del software en sí cada una se evalúa de 0 a 5. Para asignar estos valores hay que tener bastante cuidado.

Tabla 3

Tabla de valores de ajuste

Evaluar de 0 a 5 cada factor					
Sin influencia	**Incidental**	**Moderado**	**Medio**	**Significativo**	**Esencial**
0	1	2	3	4	5
		Factores			
1	Comunicación de datos				
2	Procesamiento distribuido				
3	Objetivos de rendimiento				
4	Configuración de uso intensivo				
5	Tasa de transacciones				
6	Entrada de datos en linea				
7	Eficiencia con el usuario final				
8	Actualización de datos en linea				
9	Procesamiento complejo				
10	Reusabilidad				
11	Facilidad de instalación y conversión				

12	Facilidad operacional
13	Instalaciones múltiples
14	Versatibilidad

Fuente: Pressman, 2010

3.1.20.3 Indicadores.

Algunas medidas podrían calcularse o estimarse a través de otras medidas, que son conocidas como indicadores y son útiles para pronosticar atributos que no logran medirse directamente o sin un modelo. Ejemplo, no es medible el tiempo de respuesta si el software no está acabado y en cuyo caso, la longitud de la ruta (path) del programa puede emplearse como un indicador para pronosticar el tiempo de respuesta a posterior antes de que el software logre ser un producto concluido.

3.1.21 Estimación del software.

Una estimación es el pronóstico más optimista con una posibilidad diferente de 0 de que sea verdad.

La estimación del software es difícil, porque el desarrollo del software es evolutivo. Se empieza con una imagen difusa de lo que se quiere elaborar, que luego se intenta ir aclarando a través del proceso.

3.1.21.1 Estimación del esfuerzo y el tiempo con KLDC.

El esfuerzo en personas - mes (PM) y el tiempo de desarrollo de un proyecto (TDP) puede calcularse con las fórmulas, propuestas por Boehm:

$PM = a \times (KLDC^{b})$ ..... [Ecuación 1]

Donde:

PM = Personas - mes

a,b = Coeficiente en base a proyecto

KLDC = Miles de líneas de código

$TDP = c \times (PM^{d})$..... [Ecuación 2]

Donde:

TDP = Tiempo de desarrollo de un proyecto

a,b = Coeficiente en base a proyecto

Los valores de los coeficientes por ejemplo son (para un proyecto de contabilidad):

Tabla 4

Coeficientes en base a proyecto

Programas	**A**	**B**	**C**	**D**
Aplicación	2,4	1,05	2,5	0,38
Apoyo	3	1,12	2,5	0,35
Sistema	3,6	1,2	2,5	0,32

Fuente: Propia

$PM = a(KLDC^{b}) = 2{,}4(5{,}2^{1{,}05}) = 13{,}55.$

$TDP = c(PM^{d}) = 2{,}5(13{,}55^{0{,}38}) = 6{,}73$ meses.

NP = PM/TDP =13,55/6,73= 2,01 = 2 personas.

Otras estimaciones del esfuerzo son las siguientes:

$PM = 5{,}2(KLDC^{0{,}91})$ (Waltson - Felix).

PM = $(5{,}5)(0{,}73)(\text{KLDC}^{1{,}16})$ (Waltson - Felix).

PM = $5{,}288(\text{KLDC}^{1{,}047})[\text{KLDC} >= 10]$ (Doty).

Como puede verse, los resultados van a salir muy diferentes, hay que calibrarlos de acuerdo a la naturaleza de cada aplicación.

3.1.21.2 Estimación del esfuerzo con PF.

El esfuerzo en personas - mes (PM) y el tiempo de elaboración de un proyecto (TDP) puede calcularse con las fórmulas:

PM = $-13{,}19 + 0{,}0545 \times \text{PF}$ (Albrecht y Gaffney).

PM = $60{,}62 \times 7{,}28(10^{-8})(\text{PF}^3)$ (Keremer).

PM = $587 + 15{,}12 \times \text{PF}$ (Matson, Mellichamp, Barnett).

Como puede verse, los resultados van a salir muy diferentes, hay que calibrarlos de acuerdo a la naturaleza de cada aplicación. Nosotros vamos a aplicar la primera sugerida por Albrecht y Gaffney.

3.1.21.3 Estimación de la productividad.

La productividad se define por la fórmula:

Productividad = KLDC/PM

Productividad = PF/PM

Ejemplo: Para el mismo proyecto de contabilidad con 5,2 KLDC.

- Productividad = KLDC/PM = 5,2/ 13,55 = 0,3838, es decir 383,8 líneas de código por persona-mes.

Ejemplo: Para un proyecto de Planillas con 5,61 PM.

- Productividad = PF/PM = 344,96 / 5,61 = 61,49, es decir 61,49 puntos de función por persona-mes.

3.1.21.4 Estimación por analogía.

Esta técnica es útil si se han elaborado otros proyectos en el mismo dominio de la aplicación.

3.1.21.5 Técnica de estimación con LDC y PF.

En esta técnica la descomposición funcional es esencial. Se construye una tabla básica en la que se trabaja con datos históricos.

3.1.22 Estimación del esfuerzo: matriz de costos.

Es una de las técnicas más usadas para calcular el costo de un proyecto. La estimación del esfuerzo se hace delimitando las funciones del software conseguidas del ambiente. Para cada función debe efectuarse tareas de ingeniería del software.

3.1.23 Cocomo (modelo de costo constructivo).

El Cocomo (constructive cost model) es el modelo de estimación de costos porque es el más completo y detalladamente documentado de los modelos de estimación del esfuerzo (conte). Se basa en el tamaño del software en KLDC y en base a él se calcula:

- Esfuerzo = PM en personas - mes.
- Tiempo de desarrollo del proyecto = TDP en meses
- Costo total del personal = CTP en dólares.

3.1.24 Equivalencia entre KLDC y PF.

La tabla de equivalencia entre KLDC y PF puede sernos de bastante utilidad, para hacer estimaciones del esfuerzo PM y el tiempo de desarrollo del proyecto para un proyecto del que no tenemos información sobre líneas de código.

Ejemplo: para un proyecto de administración de una empresa constructora donde hemos obtenido 344,96 PF. Para estimar PM y TDP podemos hallar su equivalencia en KLDC. Suponemos que se va a utilizar el Visual Basic como lenguaje:

- Por la tabla tenemos LDC/FP = 32, entonces LDC $032 \times 344{,}96$ = 11038,72; es decir 11,039 KLDC.

Entonces

- PM = $a(\mathrm{KLDC}^{b}) = 2{,}4(11{,}039^{1{,}05})$ = 29,87.
- TDP = $c(\mathrm{PM}^{d}) = 2{,}5(29{,}87^{0{,}38})$ = 9,09 meses.
- NP = PM/TDP = 29,87/9,09 = 3,28 3 personas.

Tabla 5

Tabla histórica de lenguajes de programación: Muestra cantidad de líneas de código divididos por puntos de función.

Número	Leguaje o entorno	LDC/PF	NRO	Leguaje o entorno	LDC/PF
1	4GL	40	14	Hoja de cálculo	6
2	ADA 95	53	15	Java	53
3	APL	32	16	Modula 2	80
4	Visual Basic	32	17	Oracle 2000	23
5	C	128	18	Paradox	36
6	C++	29	19	Pascal	91
7	Clipper	19	20	Power Builder	16
8	Delphi	29	21	Prolog	64
9	Ensamblador (Macro)	213	22	Visual C++	34
10	Forth	64	23	Cobol	106
11	Fortran 77	105	24	Visual Cobol	20
12	FoxPro 2,6	34	25	Smalltalk	22
13	Generador de informes	80	26	SQL	12

Fuente: Pressman, 2010

3.1.25 Tipos de métricas.

- Directas: Aquellas que se obtienen por medición directa con un instrumento de medición.
- Indirectas: Aquellas que no pueden medirse directamente, sino a través de estimaciones basándose en medidas directas.

3.1.26 Clasificación de métricas.

a. Por su orientación en:

- Orientadas al tamaño: Se usan para conseguir medidas directas del resultado y la calidad de la ingeniería de software.
- Orientadas a la función: Suministran medidas indirectas.
- Orientadas a las personas: Suministran información sobre la gente que elabora software y sobre la perspectiva humana de la efectividad de herramientas y métodos.

b. Por su cualidad en:

- Métricas de productividad: Se focalizan en el rendimiento del proceso ingeniería del software.
- Métricas de calidad: Indican cómo se ajusta el software a los requerimientos explícitos e implícitos.
- Métricas técnicas: Se concentran en las particularidades del software en sí.
- Factores que intervienen en la calidad del software.
- Operación del producto: su uso.
- Revisión del producto: su modificación.
- Transición del producto: su portabilidad.
- Corrección: ajuste del software a la función solicitada.
- Nº de defectos por KLDC.
- Facilidad de mantenimiento: facilidad para corregir un error, adecuar un programa a los cambios en las restricciones, y optimizarlo.

TMEC (Tiempo Medio Entre Cambios)

- Integridad: capacidad para soportar ataques incitados o no, contra su seguridad.

- Amenaza: posibilidad de que cierto tipo de ataque ocurra en un tiempo.
- Seguridad: posibilidad de que se puede contrarrestar un cierto tipo de ataque.
- Integridad [(1- Amenaza) × (1-Seguridad)]

3.1.27 Medidas de calidad.

Ejemplo: supongamos que en el transcurso de 72 horas tenemos las amenazas de tres virus: Michelangello, Viva Chile y Madona y de otro lado que falle la computadora y las probabilidades son del 10, 8, 5 y 4 % receptivamente. Pero contamos con antivirus y la probabilidad de combatir los tres virus son del 99, 95, 95 % y la probabilidad de reparar la falla es del 97 %.

Los términos de la sumatoria son:

[1 - 0,10 x (1 - 0,99)] = 0,999

[1 - 0,08 x (1 - 0,95)] = 0,996

[1 - 0,05 x (1 - 0,95)] = 0,9975 [1 - 0,04 x (1 - 0,97)] = 0,9988

Total: 3,9913

Se espera que en los casos que haya n amenazas y n seguridades la integridad tienda a n.

Facilidad de uso: amistad con el usuario.

- Habilidad intelectual y/o física para aprender a emplear el sistema.
- Tiempo preciso para lograr a dominar su uso.
- Incremento neto en productividad.

- Estimación subjetiva de la propensión de los usuarios hacia el sistema.
- Podemos valorar estos cuatro aspectos en una escala de 0 a 5:

Tabla 6

Valores de ajuste

Calificación	**Valor**
Nulo	0
Bajo	1
Regular	2
Bueno	3
Muy bueno	4
Excelente	5

- Eficiencia: recursos y código imprescindibles para que un programa cumpla su tarea.
- Reusabilidad: habilidad que implica utilizar módulos de un programa en distintas aplicaciones. (Modularidad, independencia del hardware y del sistema, generalidad)
- Interoperatividad: esfuerzo necesario para acoplar un sistema con otro.
- Probabilidad de fallo en demanda: Posibilidad de que un sistema se comporte de forma anómala ante una petitoria.
- Tasa de fallos: regularidad de conductas inesperadas.

3.1.28 Fiabilidad del software.

En estos tiempos de virus y curiosos informáticos la integridad del software tiene gran importancia. Este atributo evalúa y mide la habilidad de un sistema para

resistir ataques (accidentales como intencionales) contra su seguridad. El ataque se puede realizar en cualquiera de los tres componentes del software, datos, programas y documentos.

3.1.28.1 Métricas de fiabilidad.

- Probabilidad de fallo en demanda: probabilidad de que un sistema se comporte de manera rara ante una petición.
- Tasa de fallos: frecuencia de comportamientos inesperados.
- Tiempo medio de fallos (TMDF): tiempo promedio de operatividad del sistema antes que aparezca un fallo.
- Tiempo medio de reparación (TMDR): tiempo promedio para reparar un fallo.
- Tiempo medio entre fallos (TMEF): tiempo promedio que es la suma del tempo promedio de fallos y el tiempo promedio de reparación.
- TMEF = TMDF + TMDR
- Disponibilidad: probabilidad de que el sistema se halle disponible para su uso.
- Disponibilidad = [TMDF / (TMDF + TMDR)] × 100 %

3.2 Caso práctico

3.2.1 Programando en VB 2010.

En la elaboración del sistema procederemos a la creación de formulario que contengan los factores que utilizaremos para la medición del software. Lo podemos ver a continuación:

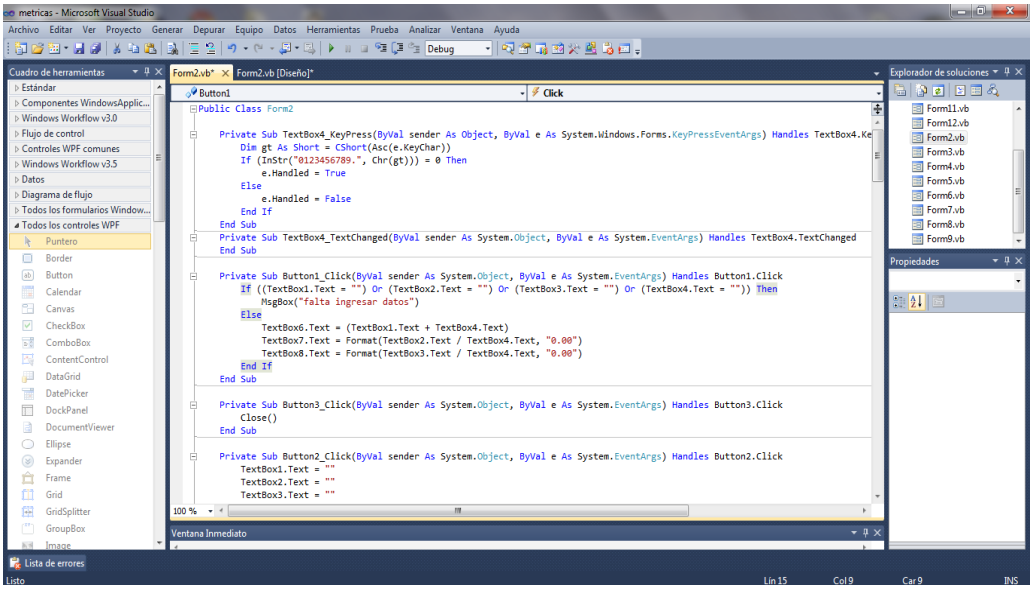

Figura 2. Identificación del contenido de los formularios y programación de los parámetros

Nota: Para su resolución se utiliza métricas orientadas por tamaño.

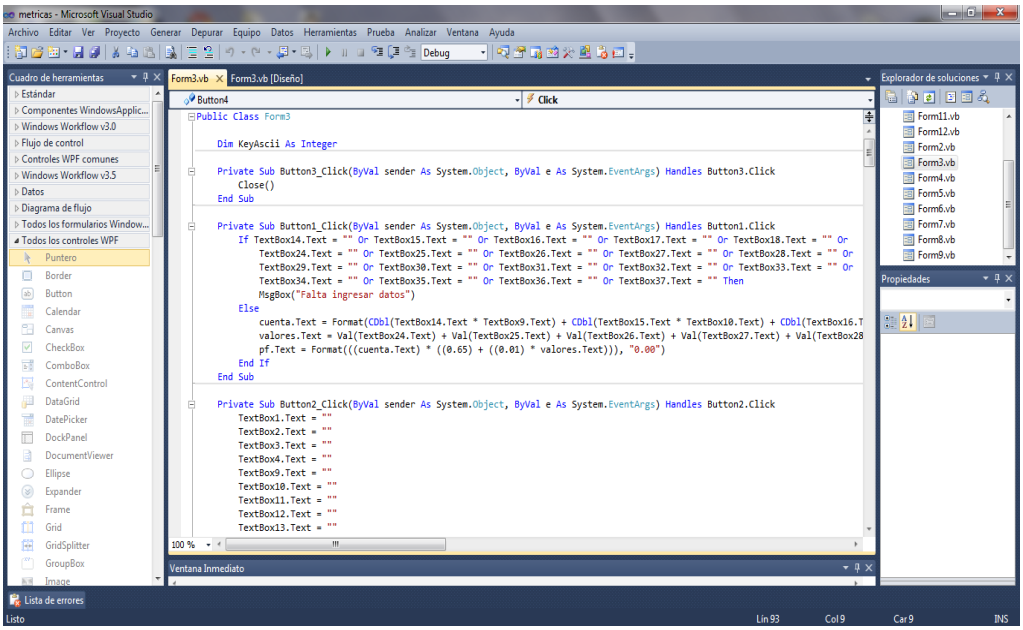

Figura 3. Codificación del formulario aplicando la fórmula para hallar puntos de función.

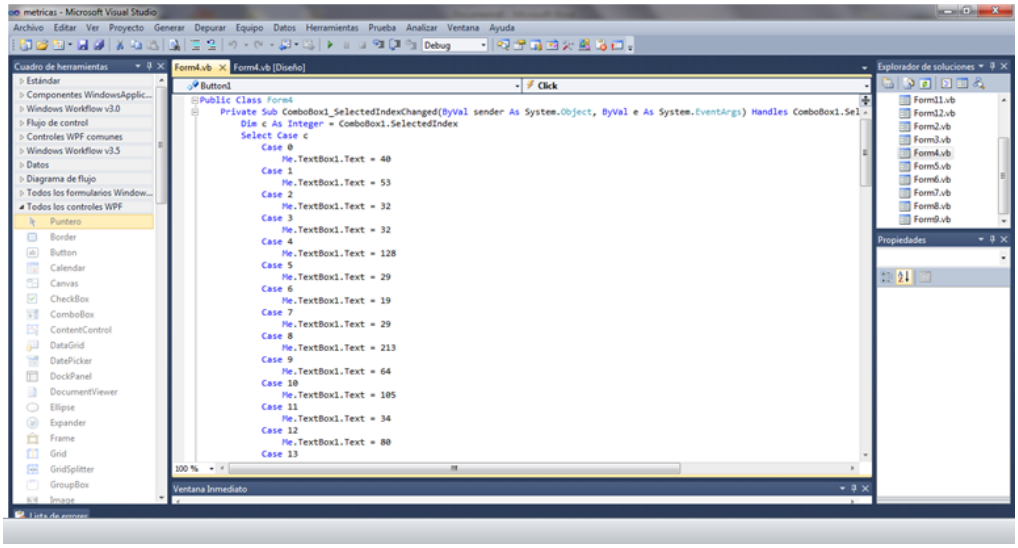

Figura 4. Desarrollo del formulario donde observamos los puntos de función

Nota: Darán lugar a la equivalencia entre ambas técnicas.

3.2.2 Tablas de instrumentos para recolección de datos.

3.2.2.1 Recursos administrativos.

Tabla 7

Recursos administrativos

Útiles de escritorio	**Cantidad**	**Unidad**	**Costo unitario**	**Costo Total**
Hojas	0,5	Ciento	10,00	5,00
Lapiceros	4	Unidad	1,00	4,00
Tableros	2	Unidad	5,00	10,00
Folder	5	Unidad	1,00	5,00

3.2.2.2 Recursos informáticos.

Tabla 8

Recursos informáticos

Dispositivos	Cantidad	Unidad	Costo Unitario	Costo Total
Laptop	1	Und.	2000,00	2000,00
Impresora	1	Und.	200,00	200,00
Usb 8gb	1	Und.	20,00	20,00

3.3 Representación de resultados.

Luego de concluir la programación, el software nos mostrará su eficiencia evaluando mediante formulas, las métricas orientadas al tamaño y asimismo la calidad del software.

3.3.1 Diseño de la aplicación.

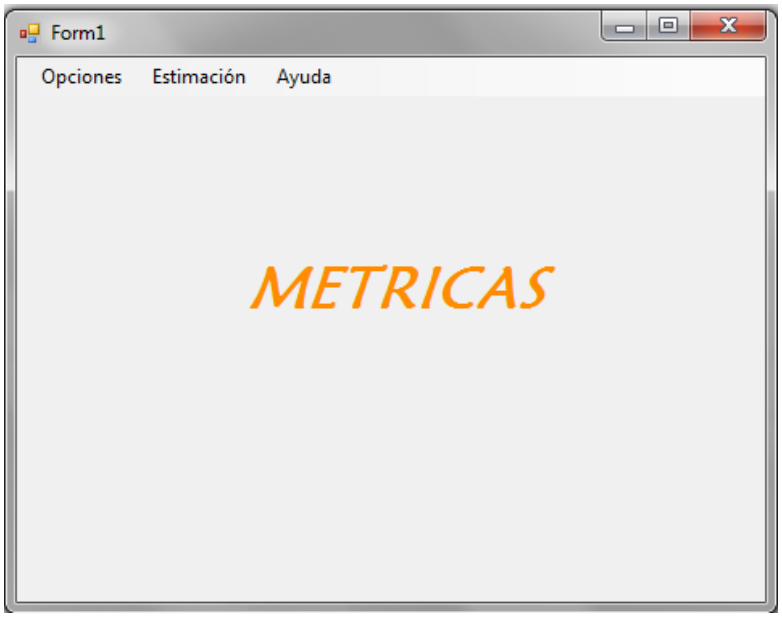

Figura 5. Formulario de ingreso al sistema métricas.

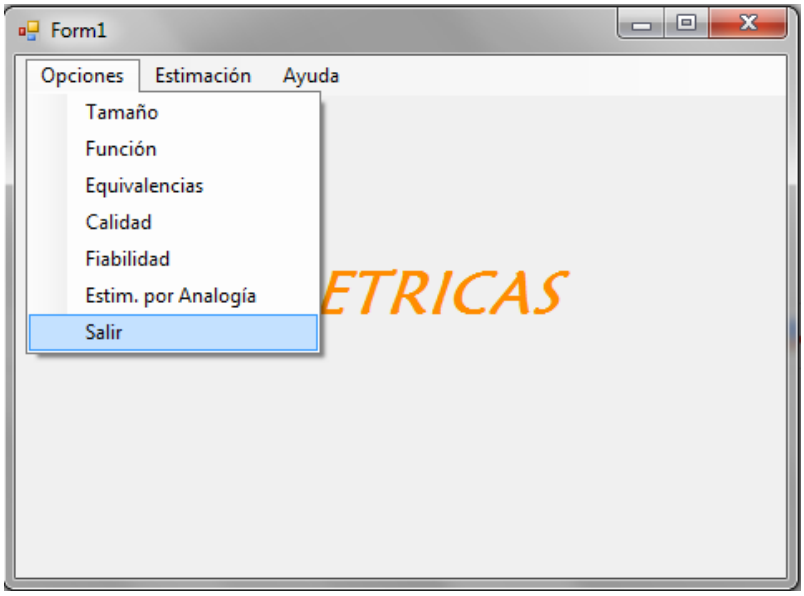

Figura 6. Opciones del sistema “métricas”

Figura 7. Opciones de estimación

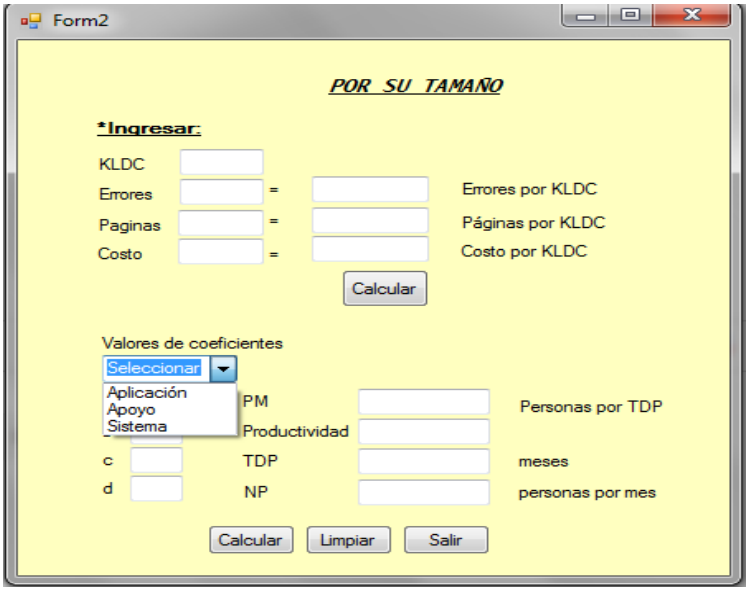

Figura 8. Diseño del formulario 2 por su tamaño

Nota: Ingresaremos los datos para calcular los errores, paginas y costo por líneas de código elaboradas, asimismo nos permitirá interpretar dichos datos para obtener la productividad, personas por mes, tiempo de duración del proyecto y la cantidad de personas por mes.

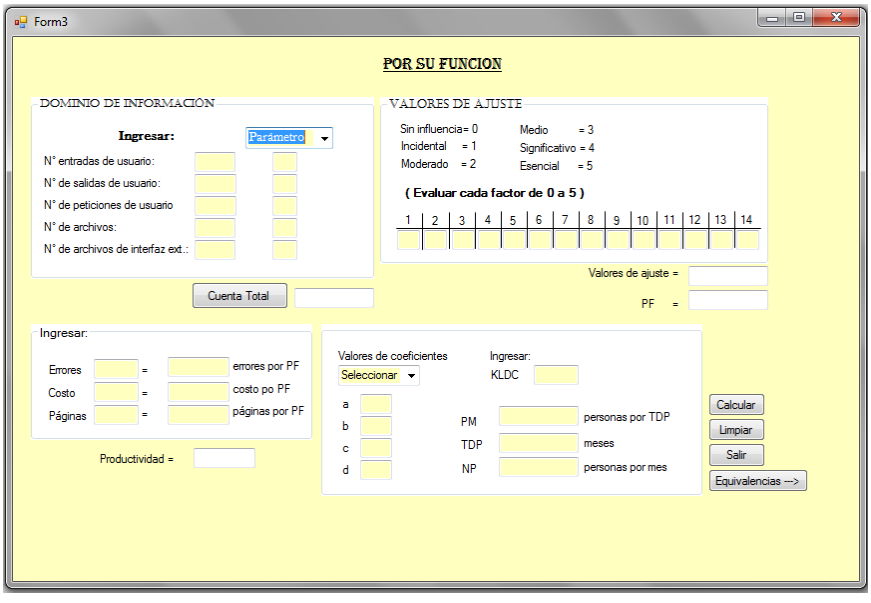

Figura 9. Diseño del formulario 3 por su función

Nota: Ingresaremos los datos necesarios para calcular los errores, páginas y costo por punto de función, asimismo nos permitirá interpretar dichos datos para obtener la productividad, por mes, tiempo de duración del proyecto y la cantidad de personas por mes.

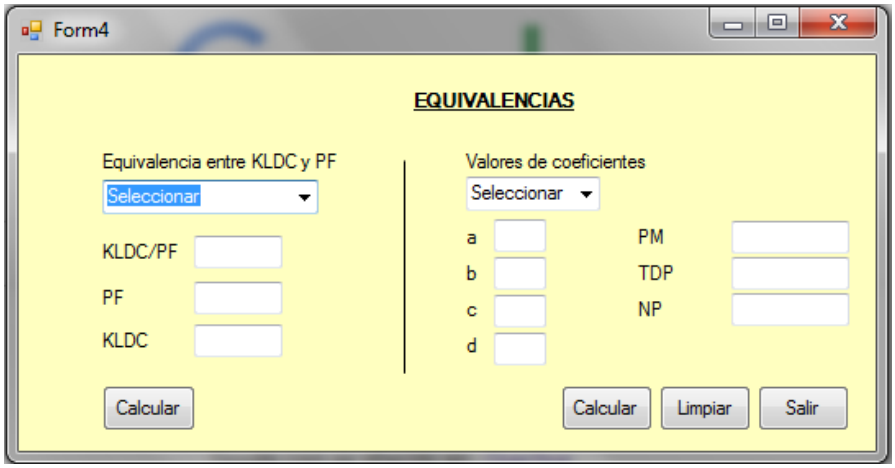

Figura 10. Diseño del formulario 4 equivalencias

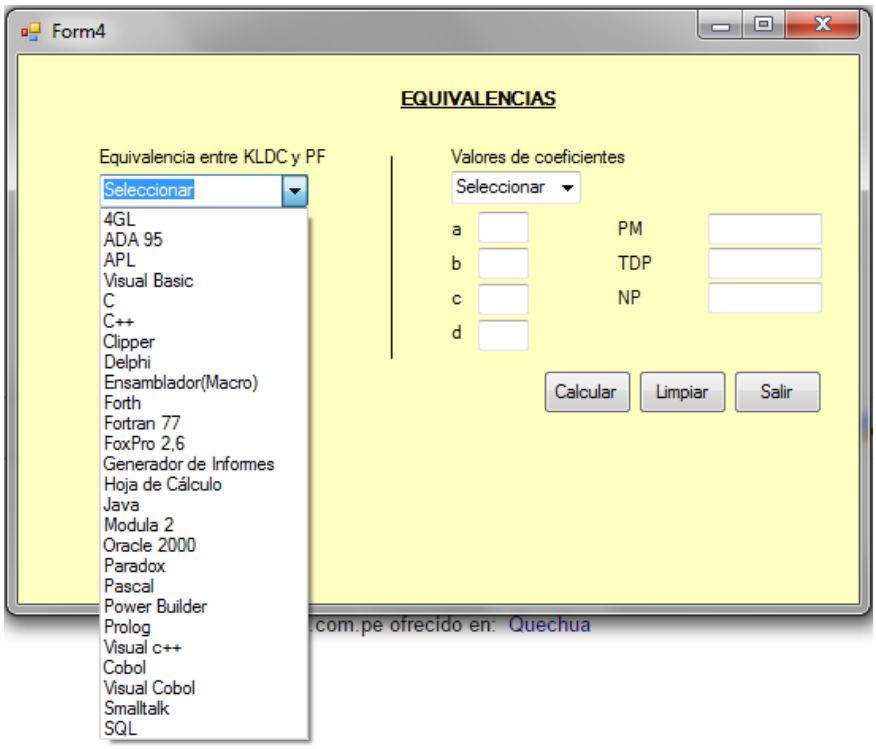

Figura 11. Formulario 4 selección del lenguaje

Nota: Esto seleccionara datos históricos de cada lenguaje de programación, asimismo seleccionar los valores de coeficientes donde se escogerán tres tipos de coeficientes de acuerdo al tipo de lenguaje a utilizar: Aplicación, Apoyo y Sistemas. Para luego hallar los datos de personas por mes, tiempo de duración del proyecto y número de personas.

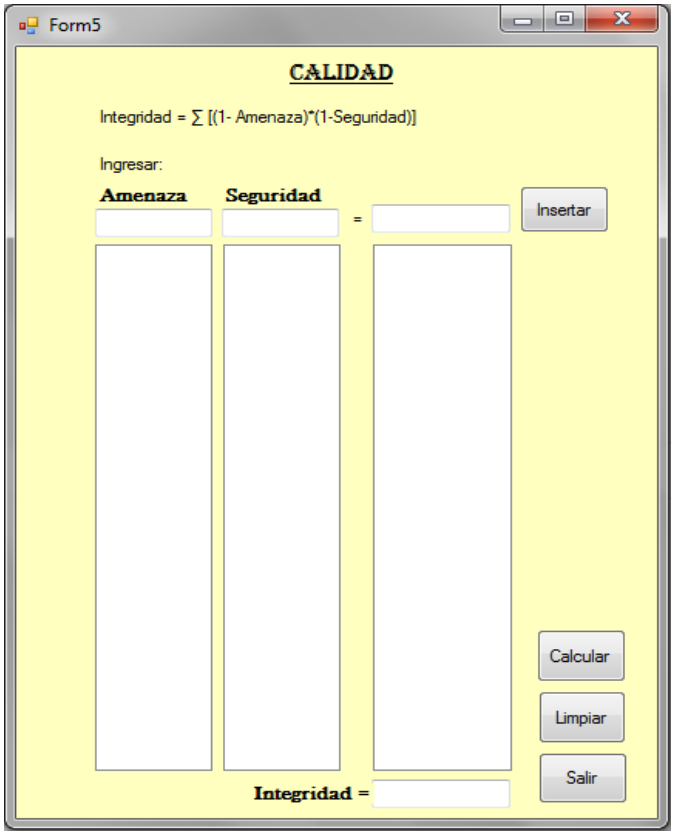

Figura 12. Formulario 5 calidad

Nota: Es este formulario calculara la integridad del software en base al ingreso de amenazas y el nivel de seguridad.

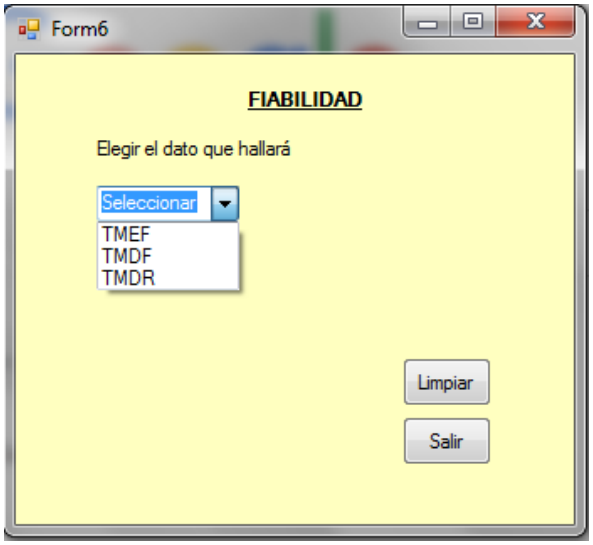

Figura 13. Formulario 6 fiabilidad

Nota: A continuación, se elegirá el tipo de dato con el cual se trabajará para encontrar la fiabilidad.

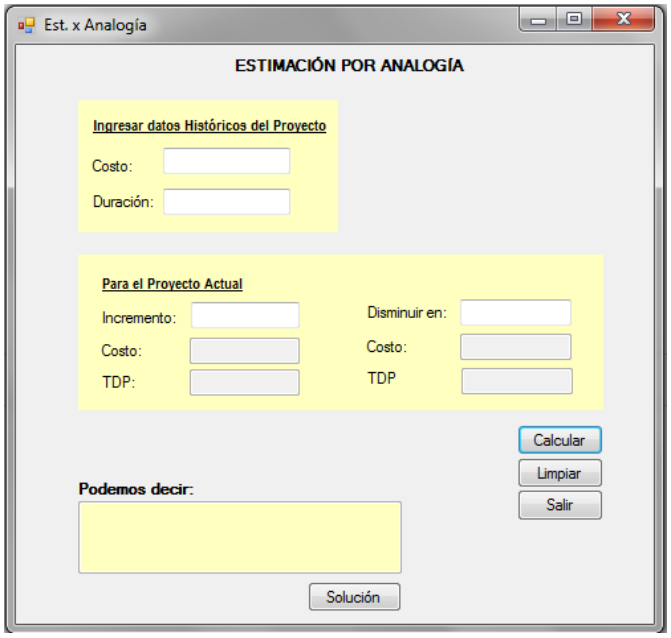

Figura 14. Formulario 7 estimación por analogía

Nota: En este formulario ingresaremos el costo y la duración del proyecto, así como su incremento o disminución.

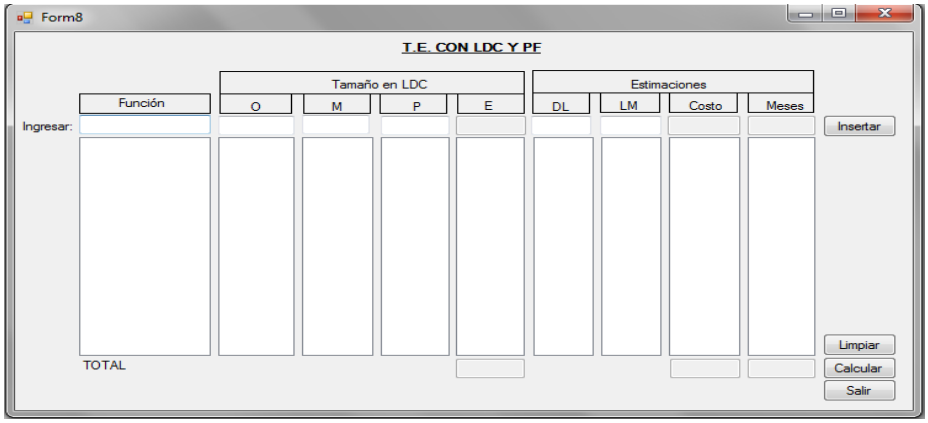

Figura 15. Diseño del formulario 8 técnica de estimación con LDC y PF

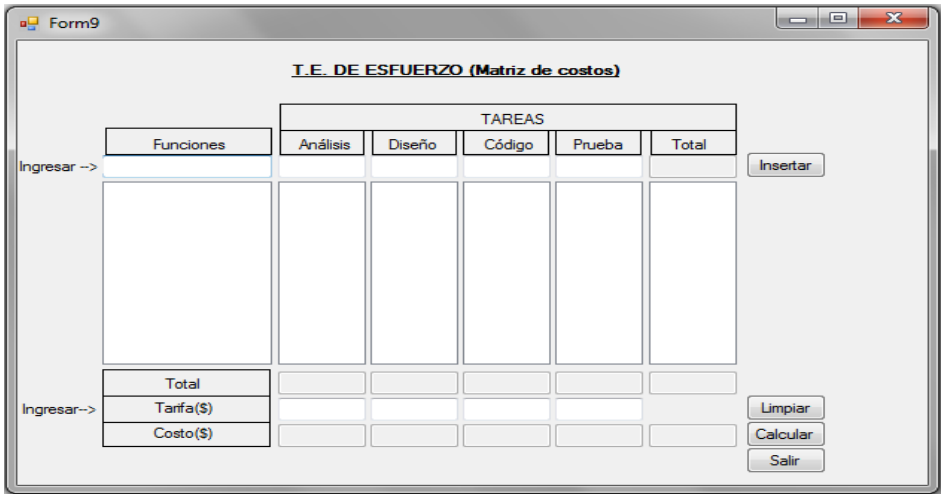

Figura 16. Formulario 9 técnicas de estimación de esfuerzo: matriz de costos

Nota: Se observa que se ha delimitado las funciones del software para cada función debe realizarse tareas de ingeniería del software como se ve en la figura.

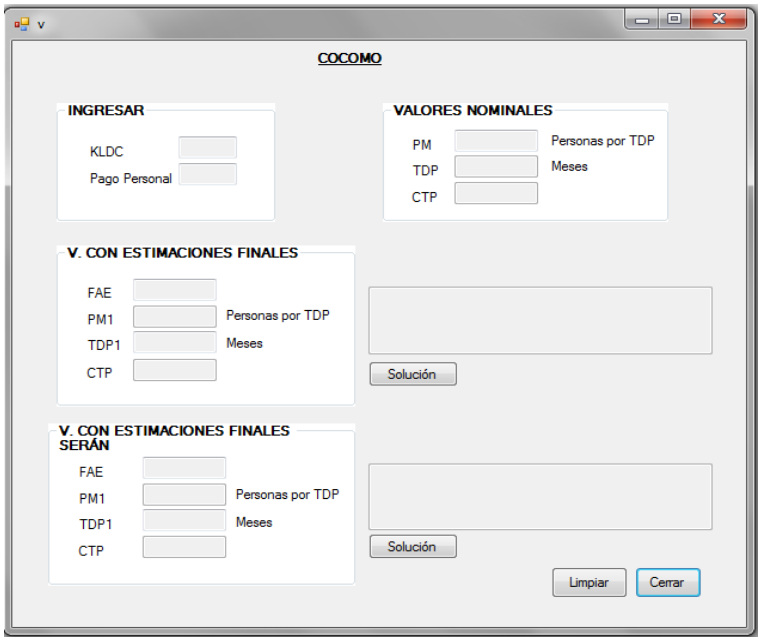

Figura 17. Formulario 10 Cocomo basada en el tamaño del software en KLDC

Nota: Existen 2 submodelos.

Modelo básico: Para calcular PM y TDP usa las fórmulas:

$PM = a(KLDC^{b})$

$TDP = c(PM^{d})$

$CTP = PM \times Pago\_personal$

Modelo intermedio: Tiene en cuenta KLDC y el factor de ajuste de esfuerzo (FAE): Estos factores pueden tomarse parcialmente o en su totalidad. De manera que PM y TDP pueden variar según el factor o factores que intervengan.

Para calcular los valores nominales de PM y TDP usaremos las fórmulas:

$PM = a(KLDC^{b})$

$TDP = c(PM^{d})$

CAPÍTULO IV

CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

Primera. El atributo de calidad es una realidad en los productos de software, para el avance tecnológico, la que se encuentra en la búsqueda de una permanente mejora de sus faenas y tareas, con el propósito de contribuir a soluciones de las inmersas insuficiencias que muestra la ingeniería de software.

Segunda. Los gestores y los ingenieros de software mediante una línea base de métricas pueden percibir de una mejor manera el trabajo y el producto que construyen.

Tercera. Al establecer resultados de evaluación de métricas de software en módulos del mismo, se crea un antecedente para evaluar futuros software's que contengan dichos módulos.

4.2 Recomendaciones

Primera. A modo de recomendación se propone la aplicación de normativas instituidas a nivel internacional, a través de la aplicación de métodos de calidad al software a desarrollar. Con ello se consigue avalar la calidad del mismo y que alcance generar su objetivo.

Segunda. Medir la calidad es algo muy relativo al criterio de cada persona, para ello es conveniente establecer rangos de valores para regular dicho criterio, es decir establecer una escala de valores que sean de forma continua y cercanos entre sí, para dar una medición a las características medidas y después valorizar el resultado en algo real.

Tercera. Luego de evaluar la calidad del software mediante métricas, este atributo implica un valor agregado al producto de software por lo que debe ser revaluado en cuanto a la relación del costo de producción y venta.

REFERENCIAS BIBLIOGRÁFICAS

Akingbehin K. (2009). *Taguchi smaller-the-best software quality metrics. 10th ACIS International Conference on Software Engineering, Artificial Intelligences, Networking and Parallel/Distributed Computing.*

Betancourt R. (2016). Métricas de software. Recuperado de http://informatica educativaysinmiedo.blogspot.com/2016/08/

Calero C.(2008). *Handbook of research on web information system quality. Editorial Advisory Board--Information Science Reference.*

Cochea, S. (2009). *Métricas de calidad en los sistemas de información*. Recuperado de https://www.dspace.espol.edu.ec/bitstream/123456789/4908/1/7708.pdf

Dolado J. (2000). *Medición para la gestión en ingeniería de software*. *Editorial RA-MA.*

Gall C.(2008). *Semantic software metrics computed from natural language design specifications. Software, IEET--IEEE.*

García O.(2009) *Material del curso de posgrado Calidad y Medición de Sistemas de Información.* Recuperado de http://alarcos.infcr.uclm.es/doc/cmsi/trabajos/Oscar%20Gomez.pdf

Heck P. (2010). *A software product certification model. Software Quality Journal.*

Hofman R.(2009). *Software quality perception. Advanced Techniques in Computing Sciences and Software Engineering. Springer.*

International Standard Organization. (2015). *Sistemas de gestión de calidad – Fundamentos y vocabulario.* Recuperado de https://www.iso.org/obp/ui/#iso:std:iso:9000:ed-4:v1:es

International Standard Organization Tools. (2015). *Indicadores de calidad*. Recuperado de https://www.isotools.org /2015/03/30/que-son-los indicadores-de-calidad/

Krall, C. (2006). *Calidad del software. métricas y fiabilidad de aplicaciones (primera parte)* (DV00103A). Recuperado de https://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=198:calidad-del-software-metricas-y-fiabilidad-de-aplicaciones-1a-parte-dv00103a&catid=45&Itemid=164

Mora, C. (2011). *La calidad del servicio y la satisfacción del consumidor.*

Olivares, M. (2013). *Calidad*. Recuperado de http://clasecalidadeinnovaciontecnologica.blogspot.com/2011/01/calidad.html

Pérez, J. (2008). *Software*. Recuperado de https://definicion.de/software/

Pressman, R. (2010). *Ingeniería del software un enfoque práctico.7ed.*

Pressman, R. (2008). *Ingenieria de Software, Un enfoque practico. 6 Edicion. Barcelona: Editorial McGraw-Hill.*