MACMHA - Marco Conceptual De Atributos, Métricas Y Heurísticas De Calidad De Software Para La Valoración Del Producto Software Orientado A Objetos

Liliam Paola Bolaños Rengifo

Manuel Alejandro Navia Porras

UNIVERSIDAD DEL CAUCA

FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES

PROGRAMA INGENIERÍA DE SISTEMAS

Grupo de I+D en Tecnologías de la Información

POPAYÁN

2010

MACMHA - Marco Conceptual De Atributos, Métricas Y Heurísticas De Calidad De Software Para La Valoración Del Producto Software Orientado A Objetos

Liliam Paola Bolaños Rengifo

Manuel Alejandro Navia Porras

Trabajo de investigación para optar al título de Ingenieros de Sistemas

Director:

Mag. Jorge Jair Moreno Chaustre

UNIVERSIDAD DEL CAUCA

FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES

PROGRAMA INGENIERÍA DE SISTEMAS

Grupo de I+D en Tecnologías de la Información

POPAYÁN

2010

AGRADECIMIENTOS

A Dios por permitirnos llegar hasta aquí.

A nuestras familias por su gran apoyo y comprensión en cada una de las etapas de la vida.

Al Mag. Jorge Jair Moreno, por su orientación, paciencia y ánimo brindados durante el transcurso del proyecto.

A las empresas que colaboraron con el proyecto, pues sin su colaboración no hubiese sido posible la culminación de este trabajo.

A nuestros compañeros por su comprensión y apoyo en cada momento durante la elaboración de este proyecto

A la Universidad del Cauca y a su Programa de Ingeniería en Sistemas por forjarnos como personas y profesionales que buscan la excelencia con el propósito de mejorar la calidad de vida de la sociedad.

TABLA DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Para la industria software existen algunas preocupaciones, que determinan la referida "crisis del software", centrada básicamente en 3 aspectos: construcción de software de calidad, satisfacción de la demanda y finalmente entregas rápidas y a bajo costo. La calidad hace parte esencial de muchas de las definiciones de la ingeniería del software, como disciplina de la ciencia informática, tal como lo afirma Pressman [1]: "disciplina de las ciencias de computación que aplica y desarrolla métodos, metodologías y herramientas para favorecer el desarrollo y mantenimiento de software de calidad". Para algunos autores [1][2][3] la calidad de software hace referencia al conjunto de atributos deseables que posee un producto software, éstos son medibles (cuantitativa o cualitativamente), permitiendo hacer comparaciones para conocer si las expectativas del usuario o cliente son cumplidas. Dichos atributos pueden dividirse en internos y externos, donde los primeros (cohesión, acoplamiento, etc.) pueden presentar influencias sobre los segundos (usabilidad, funcionalidad, etc.), por el momento relacionados de manera empírica [4][5].

En el mismo sentido, existen diversos modelos y estándares [6] de calidad de software, que poseen múltiples interpretaciones no relacionadas de forma clara sobre este concepto [4][5]. Lo anterior implica de una parte la sobresaturación de información para aquellos que tienen recursos, tiempo y habilidades suficientes como para "usar" adecuadamente un modelo y por otra parte produce des-orientación en otros que no cuentan con recursos, tiempo y personal calificado como para consolidar un interés mínimo en la adecuación y uso de estos modelos. Agregado a lo anterior, nuestro entorno académico-industrial está centrado en un paradigma de desarrollo mayormente caótico, el cuál promueve malas prácticas entre generaciones, donde la "calidad" es un concepto sofisticado y decorativo en los procesos de software reservado para consultores expertos, en lugar de considerarse un elemento permanente y cotidiano, que impregna toda actividad y producto de software en toda su esencia [7][8]. Localmente, factores tales como: dificultad en la obtención de la información acerca de la calidad del producto software, altos costos y acceso limitado a material de referencia especializado para su valoración [9] hacen del uso de estos modelos algo prohibitivo dadas las condiciones de tiempo en las entregas

para las empresas y la cualificación de su personal. En el mismo sentido, la ausencia de buenas prácticas en el diseño genera costos elevados a la hora de la detección y corrección de errores [10].

En consecuencia, a la luz de los resultados [7] queda manifiesto la necesidad de ofrecer a la comunidad PyME[1] y académico-investigativa la conformación de un marco conceptual que integre los aportes provenientes de diversos enfoques, al mismo tiempo que filtre sus deficiencias colectivas con el objeto de propagar las condiciones favorables mínimas para emprender una cultura de auto-reflexión y auto-crítica sobre la valoración de la calidad de los productos software. El marco conceptual propuesto, implicaría sencillez, accesibilidad y bajo costo para sus usuarios, permitiéndoles valorar la calidad en un producto de software a través de factores de calidad basados en métricas y centrados en heurísticas. Éste propone recopilar características de alto y bajo nivel de un producto software, usando métricas y heurísticas orientadas a objetos, todo lo anterior con algún nivel de instrumentación, que permita guiar las decisiones de un arquitecto basándose en los resultados obtenidos sobre la valoración de calidad de un producto.

Para lograr lo anterior, los objetivos planteados por este trabajo son:

Objetivo general

Plantear un marco conceptual para la valoración de un conjunto de atributos y métricas de calidad software relacionadas con heurísticas, el estudio y selección de dichos elementos, además de integrar los aportes hechos por diversas fuentes.

Objetivos específicos

- Conformar un compendio de atributos y métricas de calidad a partir de la revisión exhaustiva de modelos y estándares de calidad (McCall, Boehm, ISO-9126, FURPS, IEEE, Dromey, Bansiya etc.), que descarte las debilidades e integre las fortalezas halladas.
- Plantear recomendaciones cuyo carácter arquitectónico esté soportado en la determinación de las relaciones de correspondencia e influencia presentes entre atributos, métricas y heurísticas de calidad del producto software.

[1] Pequeña y mediana empresa.

- Plantear un proceso preliminar de verificación de resultados del marco conceptual presentado en este proyecto, apoyándose en muestras de empresas de software de la región.

Con la necesidad de satisfacer los anteriores objetivos, deben tenerse en cuenta aspectos generales que rodean el contexto de aplicación en cuanto al contexto de los usuarios. Es así como, el trabajo realizado por las organizaciones, a nivel industrial, incluye cada vez una mayor cantidad de usuarios y funcionalidades, a partir de la expansión del dominio de las aplicaciones, generando una mayor dificultad en la correcta adopción de modelos que permitan valorar la calidad del producto software. Debido a la complejidad de las aplicaciones y la dificultad de acceso a la información guía, muchas veces no son usados los criterios de calidad necesarios, sin embargo y debido a la competencia, las empresas están dispuestas, con mayor frecuencia, a adoptar estándares y modelos que ayuden a mejorar la calidad de sus productos software. Del mismo modo la estructura del software es considerada como importante, pues conviene contar con un patrón que guíe la selección y orden de realización de la arquitectura definida [11]. La necesidad de contar con referencias para la estructura del producto sugiere dar una mayor importancia a la arquitectura de software, a partir de modelos de referencia que apoyen el uso de un estilo arquitectónico para los diversos tipos de componentes [12].

De manera similar, en el contexto académico-investigativo, es posible observar una motivación para el estudio y mejora del producto software, no obstante en muchas ocasiones carecen de una guía que sugiera la definición de los componentes necesarios para el desarrollo de una aplicación software [13]. Generalmente los errores no son encontrados en las aplicaciones sino hasta el momento de ser liberadas; por lo tanto es necesario adoptar una guía para el análisis de atributos y métricas de calidad de software, que facilite el seguimiento de las aplicaciones a partir de la estructura del producto, en consideración con las buenas prácticas para tal fin.

En el ámbito regional es posible notar que el uso de los modelos, estándares, herramientas y otros mecanismos de calidad de software para la valoración del producto es limitado, poco conocido o usado, debido a la escasez de conocimiento o información incompleta, además del difícil acceso a los documentos que tienen relación con la valoración del producto software [9].

Por esta razón el proyecto está enmarcado en la obtención un conglomerado de características, métricas y heurísticas de calidad para la valoración del producto software a través de su estudio y centrando sus conceptos de manera más clara y entendible. Contando con la información que prestan los modelos y estándares de calidad de software, además de estudios y análisis anteriormente realizados, que nos permiten tener visión integradora para la conformación del marco conceptual.

Los resultados de este trabajo tratan de favorecer la creación de cultura en relación con: buenas prácticas, técnicas y herramientas relacionadas con la valoración del producto software, en el contexto académico de la Universidad del Cauca y las comunidades de desarrollo del software de la región, esperando impactar en mejores decisiones en relación con la calidad de los productos del contexto local.

Para la Universidad del Cauca y especialmente para el programa de Ingeniería de Sistemas el proyecto permite hacer una recopilación de los modelos de calidad de software a nivel de producto más importantes; también espera ser el comienzo de varios proyectos que tengan relación con la calidad de software y su valoración, siendo éste trabajo una iniciativa que enriquece el área de Ingeniería de Software, perteneciente al Grupo de I+D en Tecnologías de la Información – GTI del departamento de Sistemas.

La estructura del documento presentada a continuación, varía un poco con respecto al propuesto inicialmente, en el anteproyecto, por cuestiones de organización y para lograr un mejor entendimiento.

El trabajo del marco conceptual comprende seis capítulos como es mencionado a continuación: **Capítulo I - Fundamento Teórico,** presenta los conceptos considerados significativos para la realización del marco conceptual, tales como modelos de calidad y conceptos relacionados, que permitan al lector enfocarse en tema tratado por el proyecto MACMHA; ya con la ubicación del lector hacia el tema en el **Capítulo II - Modelos y criterios de clasificación** los criterios de selección de los modelos y el resultado de dicha selección son presentados; además, manifiesta los criterios para clasificar los atributos, métricas y heurísticas encontradas. Teniendo la información necesaria para el proyecto el **Capítulo III - Marco conceptual para la valoración del producto software orientado a**

objetos muestra la configuración del marco conceptual y sus elementos. Para lograrlo, los elementos encontrados son reunidos y organizados en un esquema de capas; adicionalmente el marco ofrece una vista de los conceptos según algunos autores. Finalmente, son mostradas las relaciones existentes entre los tres conceptos y una vista de las influencias entre los atributos. Relacionado a este capítulo está el **Capítulo IV - Elementos adicionales al marco conceptual,** es un complemento al capítulo anterior, retomando las relaciones existentes para plantear una relación que incluya las recomendaciones de las prácticas, agrupadas por disciplina(s). A continuación el trabajo muestra una guía de uso del marco conceptual. Además es necesario precisar la relación con el mundo real, la cual puede verse en el **Capítulo V - Experiencia con las empresas** mostrando el trabajo realizado con las empresas de la región y los resultados obtenidos tras aplicar algunos elementos del marco en dichas organizaciones. Y, finalmente, el **Capítulo VI - Conclusiones, lecciones aprendidas y trabajo futuro** muestra las conclusiones del trabajo realizado, junto con las recomendaciones para trabajos futuros. Este trabajo presenta, además, el libro de anexos en donde está todo aquel material que complementa el trabajo.

A continuación cada capítulo presentado en este documento está relacionado con los objetivos específicos definidos en la investigación:

El objetivo 1 es cumplido con lo tratado en los capítulos 1, 2 y 3, en ellos están un conjunto de definiciones y conceptos relacionados con los elementos del proyecto y la organización propuesta además de las relaciones de influencia encontradas entre los elementos.

El objetivo 2 es satisfecho con lo presentado en el Capítulo 4 acerca de las recomendaciones por disciplinas obtenidas de los elementos del marco.

El objetivo 3 es logrado en los capítulos 4 y 5 con la presentación de la guía de uso del marco conceptual y la aplicación de un compendio sacado del marco a MIPYMEs de la región sur occidental de Colombia.

CAPÍTULO I: FUNDAMENTO TEÓRICO

Los conceptos mencionados a continuación son aquellos considerados de importancia en éste trabajo y los cuales inciden en buena parte de su desarrollo.

Para la realización del proyecto es necesario conocer y comprender algunos conceptos de importancia, que sirvan para la conformación del marco conceptual, tales como: arquitectura software, modelos de calidad de software, métricas software, atributos software y heurísticas de software. Este capítulo es una inducción a los temas tratados en el proyecto y sirva como base para la comprensión del mismo, además de contribuir en la conformación del compendio.

1.1. Arquitectura Software

Según Bass[14], la arquitectura software de un programa reside en la estructura o estructuras encontradas en un sistema, ellas comprenden los elementos software, las propiedades externas de éstos y las interrelaciones que existan entre sí. Con el objeto de dar claridad al concepto es necesario explicar algunos puntos relacionados a la definición:

- **Las propiedades externamente visibles**. Son aquellas suposiciones que otros elementos pueden hacer de un elemento (e. g. manejo de fallos).
- **La arquitectura define elementos de software**. Entendido que la arquitectura abarca la información de cómo los elementos de software están relacionan unos con otros, excluyendo aquella información que no implica interacción entre dichos elementos.
- **La arquitectura comprende una o más estructuras**. Todos los proyectos con cierta categoría son divididos en unidades de implementación (modularización), las cuales nos ayudan básicamente a definir las responsabilidades en los equipos de trabajo de desarrollo, permitiendo que otras unidades de implementación puedan acceder a programas y datos.
- **Todo sistema de software tiene una arquitectura**. Esto puede afirmarse, dado que, un sistema puede mostrarse como una colección de elementos y las relaciones

entre ellos. Por desgracia la arquitectura no siempre depende de su descripción; puede existir sin documento que la describa.

- **El comportamiento de un elemento es parte integral de la arquitectura**. Permitiendo que el comportamiento de un elemento pueda ser observado o percibido desde el punto de vista de otros elementos y les permita interactuar entre sí.

Para poder analizar la calidad de un producto software es necesario conocer la arquitectura del mismo, de esta manera intenta brindar recomendaciones acerca de las características de calidad a través de las heurísticas de diseño planteadas para cada arquitectura, además al evaluar la calidad de un producto deberían tenerse en cuenta las características de calidad que son medibles, ellas pueden verse como los atributos internos de calidad del producto software, que tienen asociadas métricas para permitir la valoración de calidad de dicho producto.

1.2. Calidad de software

Para poder definir la calidad de software primero debe tenerse presente el concepto de calidad. Para ISO 9000 [2] es un conjunto de características o rasgos diferenciadores inherentes, los cuales tratan de cumplir una necesidad dada. Pressman [1] toma la calidad como un atributo o un conjunto de características que es posible medir.

Ahora, en la Tabla 1, aparecen algunas definiciones de calidad de software que dan los autores considerados en este proyecto:

Autor	Definición de calidad de software
ISO 9126 [15]	Calidad de software es la totalidad de rasgos y atributos de un producto software que le apoyan en su capacidad de satisfacer sus necesidades explicitas o implícitas
IEEE 1061-1998 [16]	La calidad de software es el grado para el cual el software posee una combinación de atributos deseados o requeridos por un sistema.
Pressman [1]	La calidad de software puede verse como “la concordancia con los requerimientos funcionales y de rendimiento explícitamente

Autor	Definición de calidad de software
	establecidos, con los estándares de desarrollo documentados y con las características implícitas que se esperan de todo software desarrollado profesionalmente"
Jetter [3]	Concibe un concepto de calidad de software, haciendo referencia a ella como algo siempre medible, y además menciona la necesidad de una base de comparación para poder hacer la medición.
Boehm [17]	La calidad para un producto software varía con las necesidades y prioridades que surgen desde la perspectiva del usuario.

Tabla 1. Conceptos de calidad de software

La definición adoptada, de acuerdo a la información que los autores antes mencionados nos brindan, es: La calidad de software esta referida al conjunto de atributos que posee un producto software, donde éstos pueden ser medidos, permitiendo hacer comparaciones para conocer si cumple con las expectativas del usuario.

Dicho conjunto proviene de una serie de modelos en los cuales son explicados, por lo que es necesario conocer qué es un modelo y cuáles son los que suministran tales atributos.

1.3. Modelos de calidad de software

Los modelos y estándares de calidad de software son herramientas que contribuyen a asegurar que un producto sea de calidad, de acuerdo a los requisitos del cliente y una serie de características que éste debe cumplir para ser usable[1][4]. Es así como existen modelos y estándares orientados a proceso y/o a producto (Tabla 2) que cooperan con la valoración de la calidad y brindan instrucciones de cómo seguir un proceso adecuado en los proyectos software.

	Año	Modelo / Estándar	Descripción
Producto	1977	McCall	El modelo de McCall [18], describe la calidad como un concepto elaborado mediante relaciones jerárquicas entre factores de calidad, basado en criterios y métricas de calidad. El término factor de calidad define características claves que un producto debe exhibir; el modelo identifica 11 factores de calidad orientados al producto y los organiza en tres puntos de vista, desde los cuales es posible contemplar la calidad de un producto: Operación, Revisión y Transición.

	Año	Modelo / Estándar	Descripción
			El aporte de éste modelo fue el establecimiento de relaciones entre métricas y características de calidad, aunque es criticado pues en algunos casos es imposible realizar medidas directas de calidad [1].
	1978	Boehm	El modelo de Boehm [17], ésta basado en el modelo de McCall[18], agregando algunas características a las ya existentes y organizándolas también de forma jerárquica, además incluye algunos criterios concernientes con el hardware y define métricas directas e indirectas para determinar el nivel de acuerdo a un criterio particular que afecte los factores de calidad [4][19]. De esta forma, el modelo propone al menos una métrica por cada característica primitiva, al mismo tiempo que promueve cada métrica como una "*medida en la cual un producto posee y exhibe un cierto atributo de calidad"* [6].
	1987	FURPS	El modelo FURPS fue desarrollado por Grady [20][19], con base en trabajos anteriores como McCall [18] y otros, propone un conjunto de factores de calidad del software que incluye cinco categorías principales y también plantea dos categorías de requerimientos: funcionales y no funcionales. La sigla FURPS proviene de los nombres en ingles de sus categorías principales: Funcionalidad (Functionality), Usabilidad (Usability), Confiabilidad (Reliability), Desempeño (Performance) y Soportabilidad (Supportability) [4]
	1988	GILB	El modelo de Gilb plantea la creación de una especificación de requisitos de calidad para cada proyecto, que deben escribir conjuntamente el usuario y el analista**.** Es un modelo que permite determinar una lista de características que definen la calidad de la aplicación [4] [21]
	1990	IEEE 610.12	El estándar IEEE 610.12 es una actualización del estándar IEEE 729-1983 [23], Para Fitzpatrick [22] tiene una relación específica con el software y es un buen candidato para un análisis más completo. Es un glosario de términos relacionados con el software, configurado como estándar.
	1993	SATC	Software Assurance Technology Center [24], de la NASA, recomienda el uso de seis de las métricas de Chimdamber y Kemerer[67] más tres métricas tradicionales. Ha desarrollado un modelo que brinda la posibilidad de producir varios proyectos en desarrollo. [4][25]
	1995	Dromey	Este modelo plantea la conexión entre las características y los atributos de calidad de un producto software; además de ofrecer una ventaja asiste en la búsqueda de los defectos de calidad, dando una guía para la construcción de un conjunto comprensible de defectos para un lenguaje en particular. El modelo no pretende ser definitivo y único para ser usado, por el contrario es un modelo empírico que está dispuesto a ser corregido y mejorado[26].
	1995	ISO 14598	Conjunto de estándares en partes de 1 al 6, da una visión general del proceso de evaluación del producto software y provee un soporte para evaluación. [27][28]
	1996	SQAE	Metodología desarrollada por MITRE para cuantificar los riesgos asociados al software [29], es a menudo utilizada

	Año	Modelo / Estándar	Descripción
			para proporcionar una evaluación global de sistemas de campo que por lo general no han sido bien comprendidos o medidos de forma adecuada [30]
	1997	Bansiya	Bansiya y Davis [31] hacen la introducción de un modelo de calidad para el diseño orientado a objetos. Utiliza altos niveles de abstracción como una forma de mapear métricas de código fuente. Aquí trata de conectar las métricas de código fuente de bajo nivel con los atributos de calidad de alto nivel, en un modelo de calidad para el diseño orientado a objetos.
	1998	GQM (Goal / Question / Metric)	Modelo propuesto por Tom Gilb, con este es posible identificar objetivos, métricas que permiten la caracterización de calidad [8]. Posee una estructura jerárquica. Un modelo de GQM es desarrollado mediante la identificación de un conjunto de objetivos de calidad y / o productividad [32].
	1998	IEEE 1061	El estándar IEEE 1061-1998 [16] establece una metodología para los requerimientos de calidad, además de identificar, implementar, analizar y validar el proceso y el producto software a través de métricas de calidad definidas. La metodología abarca el ciclo de vida de software por completo. Éste estándar no trata de ser rígido en su aplicación para la evaluación de métricas, en su lugar intenta que la organización que lo aplique pueda emplear cualquier métrica según crea más apropiada.
	2001	ISO 9126	El estándar ISO 9126 [15] fue desarrollado como intento para identificar los atributos clave de calidad del software. Aquí identifica seis atributos clave de calidad: Funcionalidad, Usabilidad, Mantenibilidad, Confiabilidad, Portabilidad y Eficiencia [1]. Éste estándar esta basado en los modelos de McCall y Boehm, siendo un modelo jerárquico con la calidad total en la cima, los seis atributos en segundo nivel y varios subfactores en el nivel más bajo [18]
	2002	QUINT2	Es una ampliación de la norma ISO 9126 hacia sitios Web, pensado para la calidad de arquitecturas software.[33][34]
	2004	PQM (Portal Quality Model)	Modelo de calidad para portales, formado por 6 dimensiones, algunas de ellas divididas en sub-dimensiones, utiliza el método GQM como base. [4][34]
Proceso	1982	Six sigma	Es una propuesta que permite mejorar proceso y productos, pretende aplicarse en dominios relacionados a tecnología de software, busca la satisfacción del usuario por medio de reducción y eliminación de defectos por medio de la probabilidad, además tiene como objetivo la reducción de costos por medio de la reducción de defectos y mejora de procesos.[4][21] [24][35]
	1991	ISO 9000-3 (TickIT)	Es un conjunto de procedimientos que permiten aplicar sistemas de gestión de calidad a las empresas de desarrollo de software. Sirve como interpretación de ISO 9001 [21][36]
	1993	ISO 15504 (SPICE)	SPICE (Software Process Improvement and Capability dEtermination). Es un modelo de madurez de procesos que busca examinar y valorar los procesos utilizados al interior de las organizaciones [4][37]

	Año	Modelo / Estándar	Descripción
	1995	ISO 12207 IEEE / EIA 12207	Presenta un marco común, para el ciclo de vida del software, que consta de procesos para adquirir y suministrar productos y servicios de software, además permite controlar y mejorar procesos y su desempeño [4][21][39][40].
	1995	Personal Software Process (PSP)	Es un proceso definido para ser usado por el ingeniero de software en forma individual. Su objetivo es guiar planeamiento y desarrollo de los módulos de software. Esta basado en los principios de mejoramiento del proceso [4][12][24][41]
	1997	Practical Software Measurement (PSM)	Define un proceso de medición práctico, orientado a la información, junto con conceptos de medición consecuentes [4][38] Tiene una relación con la norma ISO 15939 y con el modelo CMMi. Este modelo busca suministrar a los gerentes de proyectos la información cuantitativa necesaria para la toma de decisiones que impactan los costos y objetivos.
	1997	Bootstrap	Tiene como principal objetivo mejorar la capacidad de las unidades productoras de software. Trata de mejorar los procesos de software. Esta relacionado con la norma ISO 15504 [4][37]
	1999	Team Software Process (TSP)	Busca suministrar un proceso operacional que ayude a los ingenieros a hacer trabajos de calidad a través de la formación y entrenamiento de los equipos. Tiene como requisito que los ingenieros deben conocer primero PSP.[4][12]
	2002	CMMI (Capability Maturity Model Integration)	La integración de modelos de madurez de la capacidad. Es un modelo de calidad para mejora de procesos en el desarrollo y mantenimiento de los productos y servicios [38]. Esta organizado en áreas de proceso que determinan el nivel de capacidad o madurez de una organización.[21]
	2004	ISO 90003	Publicada como una concreción de la norma ISO 9126, para el sector de ingeniería del software, construyendo un puente para otras normas de tecnologías de la información relacionadas. Provee una guía en adquisición, suministro, desarrollo, operación y mantenimiento de software y servicios de soporte.[4][21][41]
	2005	ISO 25000 (SQUARE)	Requisitos de calidad de software y evaluación, es una serie de normas basadas en ISO 9126 e ISO 14598. Busca proporcionar un conjunto común de modelos de referencia, terminología, definiciones y orientación para uso práctico de normas e informes técnicos asociados.[4][42]
	2005	ISO 20000	Describe la especificación de administración de servicio. Este estándar permite a las organizaciones mejorar su capacidad de entrega en los servicios suministrados [4][37]. La norma establece una distinción entre las mejores prácticas de los procesos, que son independientes de la forma o el tamaño de la organización. Es aplicado a los proveedores de servicios grandes y pequeños, y los requisitos para las mejores prácticas de gestión de procesos de servicio no cambian de acuerdo a la forma de organización que proporciona el marco de gestión en el que los procesos son seguidos.[43]

Tabla 2. Modelos y estándares de calidad de software

Este trabajo ha tomado a consideración los modelos y estándares de calidad para producto software, este tema es tratado con mayor profundidad en los capítulos siguientes.

1.4. Medición en el software

La noción de medición utilizada en el software necesita otros conceptos relacionados para su comprensión, tales como: medición y medida, métrica y métrica de calidad, atributo y atributo de calidad y heurísticas; estos son explicados en los siguientes ítems:

1.4.1. Medición y medida

Para [16] la medida es una forma de establecer o cuantificar un valor por la comparación de éste a una norma, y la medición es el acto o proceso de asignar un número o categoría a una entidad para describir un atributo de dicha entidad.

1.4.2. Métrica y métrica de calidad

En [23] una métrica es una medida cuantitativa del grado en el que un sistema, componente o proceso posee un atributo. Del mismo modo comprende el concepto de métrica de calidad como una función cuyas entradas son datos software y cuya salida es un valor numérico único, que puede ser interpretado como el grado en el cual el software posee un atributo de calidad dado; en otro sentido [16] difiere en la definición de métrica de calidad que ofrece [14], pues para éste el valor de la función puede ser interpretado como el grado en el cual el software posee un atributo dado que afecta su calidad. También [17] define una métrica como el grado en el cual un producto posee y exhibe una cierta característica. Para ser utilizadas de manera correcta, las métricas de calidad pueden clasificarse según su taxonomía (Por característica o atributo del software a medir, Por etapa del ciclo de vida en que es medido y Por el nivel de [2]granularidad en el que es medido), según su tipo (Directas e indirectas), además poseen unas propiedades ideales (simplicidad, objetividad, facilidad, robustez, validez) [14].

[1] Nivel de detalle de un componente

La métrica de calidad puede considerarse como una medida cuantitativa, esperado valorar el grado en el cual está clasificado un atributo software, y además analizar si este afecta la calidad del producto evaluado.

1.4.3. Atributo y atributo de calidad

Un atributo es una característica o propiedad ya sea física o abstracta de un ítem o entidad que es medible. Un atributo de calidad puede referirse como una característica software, que afecta los ítems y/o factores de calidad (un factor es un atributo orientado a la gestión de software que contribuye a su calidad) [16][23]. Al igual que las métricas, los atributos de calidad pueden clasificarse según su tipo (internos y externos) para facilitar su medición [18], además dependiendo del modelo y teniendo en cuenta que la calidad puede verse como una mezcla de factores que varían a través de diferentes aplicaciones y clientes que las pidan, los atributos pueden ser divididos en factores y sub factores [1].

1.4.4. Heurísticas

Teniendo en cuenta algunos puntos de vista de varios autores, podemos decir que una heurística es:

- Desde una perspectiva general, un conjunto de guías de desarrollo, que describen una estrategia para la aplicación satisfactoria de un método [45].
- Desde otra perspectiva, referida a reglas aproximadas tomadas como base para la solución de un problema o contribuirán a reducir el tiempo en la búsqueda de la solución [46].
- Las heurísticas también pueden ser atajos para la solución de nuevos problemas que incluyen dificultades previamente resueltas por expertos, quienes encuentran patrones en las nuevas situaciones y a partir de ellos logran responder apropiada y rápidamente [47].

Entonces podría pensarse que una heurística es una serie de observaciones o recomendaciones, para contribuir a resolver un problema, de manera que invierta menos tiempo en encontrar la solución. Teniendo en cuenta lo anterior están propuestas un gran

número de heurísticas de diseño orientado a objetos, cada autor plantea dichas heurísticas de manera distinta y con diferentes características, a continuación son mencionados algunos aspectos presentados respecto al término heurística:

- Riel [82] cataloga 68 heurísticas de diseño clasificadas en 8 diferentes grupos: Clases y Objetos, Topologías orientadas a la acción vs. aplicación orientada a objetos, la relación entre clases y objetos, relaciones de herencia, herencia múltiple, relaciones de asociación, especificación de clase y comportamiento de datos, y diseño físico orientado a objetos. Para cada una de las heurísticas señaló que tiene nombre, resumen y ejemplo del problema, además de sugerir una aproximación a la solución del mismo.
- TOAD (Teaching Object-Oriented Analysis and Design) [47], es un prototipo para evaluar diseños orientados a objetos. Para cada heurística plantea un nombre, un propósito (por qué las heurísticas deberían ser usadas), propiedades de las heurísticas, consecuencias de éstas (cuáles son las ventajas y desventajas de las heurísticas), entre otros. Además menciona una motivación con que podría estar relacionada la aplicación de la heurística. Las heurísticas de TOAD son declaradas como un tipo de evaluación informal para categorizar el flujo del diseño orientado a objetos.
- MeTHOOD (Measures, Transformation Rules, and Heuristics for Object-Oriented Design). Éste modelo documenta 20 heurísticas de diseño orientado a objetos. Para cada una de las métricas descritas aquí [47], plantearon un nombre de las heurísticas, una pequeña descripción, una definición, posición en el ciclo de vida, un ejemplo, heurísticas relacionadas, reglas de chequeo, heurísticas violadas además de una justificación y efectos de la violación de las heurísticas. El modelo agrupa las heurísticas en 3 diferentes grupos: heurísticas verificables, heurísticas no-verificables y heurísticas implicadas. En las heurísticas verificables, un algoritmo o función de chequeo puede ser aplicado a un fragmento del diseño que evalúa la certeza o falsedad de la verificación, y además chequea independientemente las violaciones. Las heurísticas que no pueden ser chequeadas automáticamente para un fragmento del diseño dado, son conocidas como heurísticas no-verificables. Las heurísticas implicadas hacen referencia a recomendaciones para optimizar el valor de una medición y que no tienen formula de chequeo consistente.

Los conceptos definidos en el numeral 1.4 pretenden aplicarse a un entorno específico; para éste caso micro empresas dedicadas al desarrollo de software, MIPYMEs. A continuación es especificada una definición de esté término y su importancia en el comportamiento de la economía del país.

1.5. MIPYMEs

En este numeral trataremos de definir lo que es una MIPYME, con el objeto de conocer el tipo de empresas que son consultadas y colaboran con el proyecto.

1.5.1. Definición

Por pequeña y mediana empresa[3] es entendida toda unidad de explotación económica, realizada por persona natural o jurídica, en actividades empresariales, agropecuarias, industriales, comerciales o de servicios rurales o urbanos, que responda a dos de los siguientes parámetros [49]:

Mediana empresa:

a. Planta de personal entre cincuenta y uno (51) y doscientos (200) trabajadores, o
b. Activos totales por valor entre cinco mil uno (5.001) a treinta mil (30.000) salarios mínimos mensuales legales vigentes.

Pequeña empresa:

a. Planta de personal entre once (11) y cincuenta (50) trabajadores, o
b. Activos totales por valor entre quinientos uno (501) y menos de cinco mil (5.000) salarios mínimos mensuales legales vigentes.

En esta definición no esta el concepto de microempresa, pero debido a las características de la Mediana empresa es posible pensar en la siguiente definición:

Microempresa [50]:

a. Planta de personal no superior a los (10) trabajadores;

[3] Artículo 2, Ley 905 de 2004

b. Activos totales por valor inferior a quinientos uno (500) salarios mínimos mensuales legales vigentes.

1.5.2. Importancia

De acuerdo con el documento Conpes 3484[4], estableció que: "*las microempresas y las PyMEs son actores estratégicos en el crecimiento de la economía, la transformación del aparato productivo nacional, y el mejoramiento de la posición competitiva del país*".

Según los estudios realizados por el DANE, para el censo general de 2005, la estructura de empresa Colombiana esta centrada principalmente en MIPYMEs, siendo las mayores generadoras de empleo, como lo muestran la Figura 1y Figura 2.

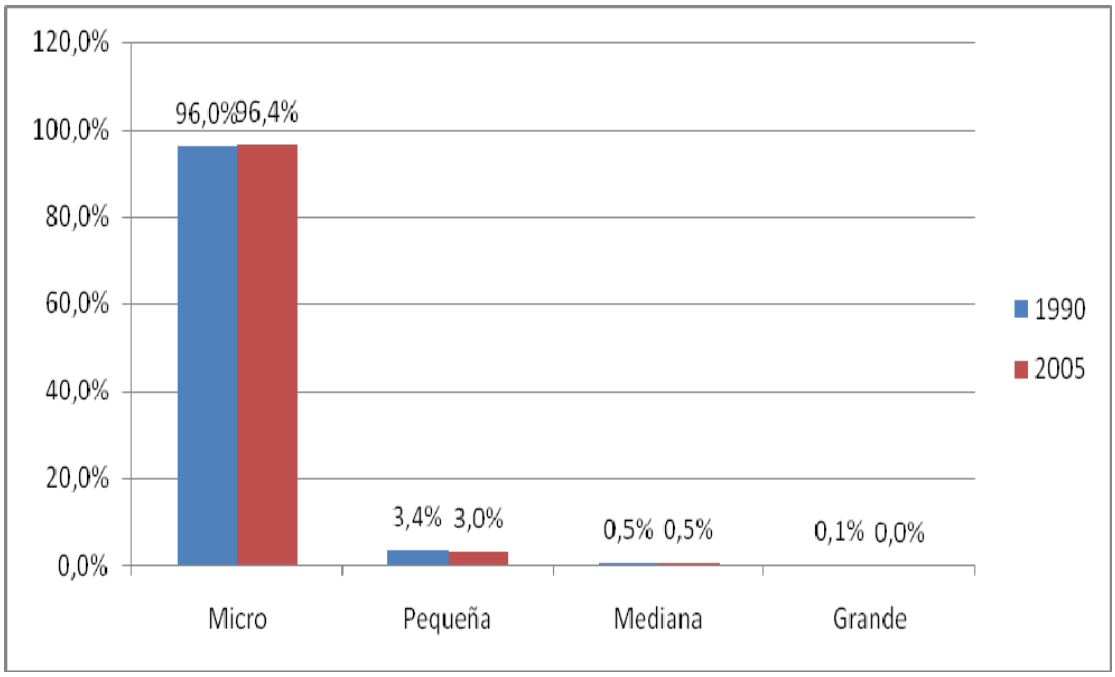

Figura 1. Porcentaje de establecimientos (Conpes 3484)

[4] Documento Conpes 3484 de 2007 (Política Nacional para la Transformación Productiva y la Promoción de las Micro, Pequeñas y Medianas empresas: un esfuerzo público-privado).

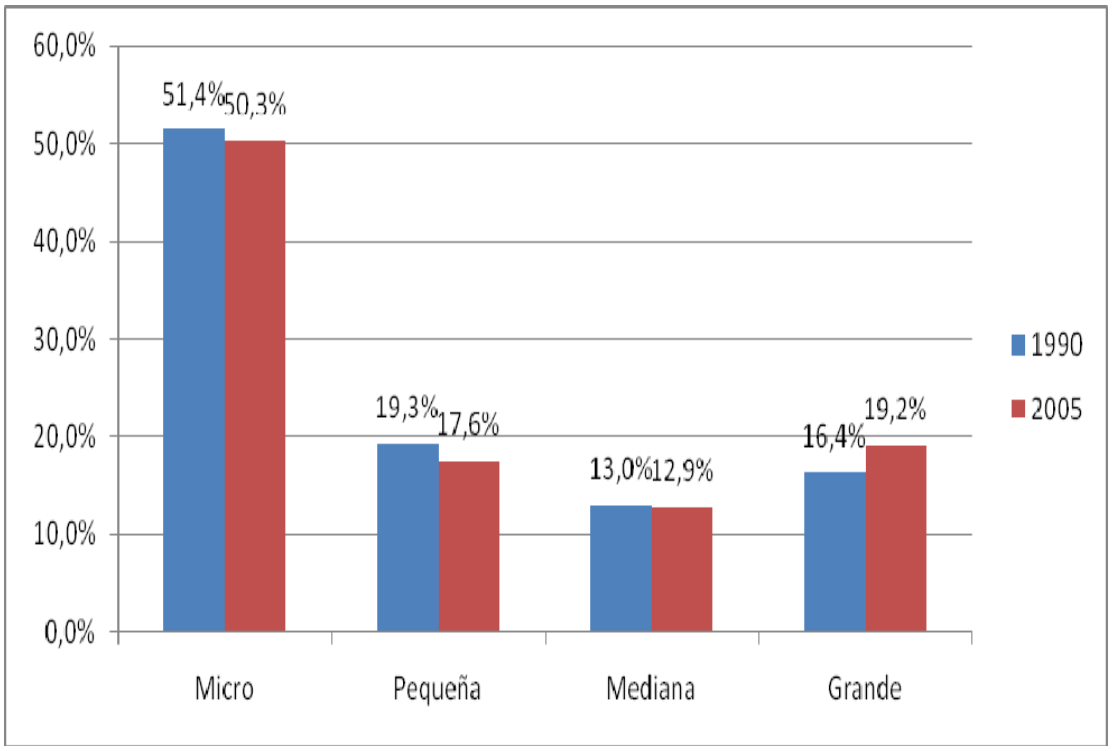

Figura 2. Porcentaje de personal ocupado por las empresas (Conpes 3484)

Según estos estudios la gran industria ocupa 40.000 personas más en el 2005 que 15 años atrás, la mediana empresa perdió 7.000 empleos en ese periodo, la pequeña empresa gano 30.000 empleos, y la micro empresa en ese mismo periodo ganó 191.000 empleos.

Las definiciones y conceptos mostrados en este capítulo, son los pilares sobre los que está apoyado el desarrollo de este proyecto. En capítulos posteriores es mostrado el desarrollo de acuerdo a la intención del presente proyecto. Para seguir con esta intención, inicialmente es tratado el tema de los modelos y estándares de calidad y la aplicación de criterios de selección a los mismos para obtener aquellos que satisfacen ciertas condiciones que rodean el desarrollo de este proyecto.

CAPÍTULO II: MODELOS Y CRITERIOS DE CLASIFICACIÓN

Basado en los diferentes conceptos mencionados en el capítulo anterior, el presente intenta recopilar aquellos indicados por los autores más reconocidos. Cada uno de ellos presenta una forma diferente de abordar el tema de la calidad del software, a partir de elementos similares a los que pretende tratar este proyecto (atributos, métricas y heurísticas). Este capítulo toma dicha información como base de los criterios de selección de los modelos y los criterios de clasificación para atributos, métricas y heurísticas de calidad, tal que sirvan de apoyo a lo planteado aquí.

2.1. Los modelos de calidad para producto software

Antes de comenzar a trabajar con los atributos, métricas y heurísticas de calidad, es de importancia estudiar los modelos de calidad para producto software, su selección y sus características generales.

2.1.1. Criterios de selección para modelos de calidad del producto software

Con el fin de centrar algunos conceptos de calidad que tengan relación entre sí, y a su vez pertenezcan a la clasificación de modelos o estándares de calidad del producto software, es necesario tener en cuenta algunos criterios de selección que sirvan para elección de aquellos modelos que pueden ser de interés para este trabajo. Los criterios presentados a continuación tienen su base en trabajos anteriores [51].

- C1: *Disponibilidad*: grado en que es posible acceder a la información existente. Es referida a la facilidad de obtener la información.
  - 1: la información no está disponible al público en general
  - 2: Hay disponibilidad de algunos documentos pero es limitado el acceso.
  - 3: Existe información suficiente disponible para ser usada.

- C2: *Claridad*: Grado en que el modelo es presentado y si posee mecanismos explicativos sobre su uso. Es referida a qué tan sencillo puede ser entender el modelo; influyen factores como: estructura, idioma y presentación del modelo.

1: El modelo no es claro o dificulta su entendimiento, no posee mecanismos de ayuda sobre el modo de emplearlo.

2: El modelo presenta su información en forma clara, sin embargo no posee mecanismos de ayuda sobre el modo de emplearlo.

3: El modelo presenta su información en forma clara, posee mecanismos explicativos acerca de su modo de empleo.

- C3: *Adaptabilidad*: Grado en que el modelo posee la capacidad de adaptarse a distintas situaciones dependiendo del producto al cual aplicar.

  1: El modelo no es adaptable. Es presentado de forma rígida para su uso.

  2: El modelo puede ser adaptado pero exige ciertas reglas a seguir.

  3: El modelo permite ser adaptado.

- C4: *Completitud*: Grado en el que el modelo describe todas sus partes en su totalidad sin dejar por fuera información importante. Un modelo completo es aquel que posee descripción de atributos, métricas y mecanismos de ayuda para llegar a la medición.

  1: El modelo no menciona toda la información necesaria. Esta muy incompleto

  2: El modelo describe medianamente sus componentes, sin embargo deja algunos elementos por fuera. Está parcialmente incompleto.

  3: El modelo describe todas sus partes. Esta completo.

Adicional a los 4 primeros criterios de selección surge un 5º que es de interés para este trabajo y tiene relación a si es o no un modelo para producto, éste es descrito a continuación.

- C5: *Área de aplicación*: aplicabilidad del modelo a las diferentes áreas de calidad del software.

  1: Modelo de proceso, metodología o estándar (no incluye modelo)

  2: Puede ser modelo de proceso y producto al mismo tiempo.

  3: Modelo para producto software

Para cada modelo o estándar existe la posibilidad en que no exista el caso en el que no aplica el criterio. Si esto sucede, la notación NA es utilizada y no suma puntuación

La Tabla 3 muestra los resultados del estudio realizado a los modelos de calidad mencionados en el capítulo 1, sección 1.3, teniendo en cuenta los criterios de selección mencionados. Para ampliar la información puede referirse al Anexo A

Modelo / Estándar	**Criterios**					**Total**
	C1	**C2**	**C3**	**C4**	**C5**	
McCall	2	2	2	2	3	11
Boehm	2	2	2	2	3	11
FURPS	1	2	3	1	3	10
GILB	2	NA	NA	NA	3	5
IEEE 610.12	2	NA	NA	NA	1	3
SATC	2	2	2	3	2	11
Dromey	1	2	2	2	3	10
ISO 14598	1	1	NA	NA	2	4
SQAE	2	2	2	2	3	11
Bansiya	2	2	2	2	3	11
GQM (Goal / Question / Metric)	2	1	3	1	2	9
IEEE 1061	2	NA	NA	NA	1	3
ISO 9126	2	2	2	2	3	11
QUINT2	1	2	2	2	3	10
PQM (Portal Quality Model)	2	1	3	1	2	9
Six sigma	2	NA	NA	NA	1	3
ISO 9000-3 (TickIT)	2	NA	2	NA	1	5
ISO 15504 (SPICE)	2	2	1	NA	1	6
ISO 12207 o IEEE / EIA 12207	2	2	2	NA	1	7
Personal Software Process (PSP)	1	NA	NA	NA	1	2
Practical Software Measurement (PSM)	1	NA	NA	NA	1	2
Bootstrap	1	NA	NA	NA	1	2
Team Software Process (TSP)	1	NA	NA	NA	1	2
CMMI (Capability Maturity Model	2	3	1	1	1	8

Modelo / Estándar	Criterios					Total
	C1	C2	C3	C4	C5	
Integration)						
ISO 90003	2	NA	NA	NA	1	3
ISO 25000 (SQUARE)	2	2	2	NA	1	7
ISO 20000	2	NA	NA	NA	1	3

Tabla 3. Aplicación de criterios a los Modelos y Estándares de calidad software

Teniendo en cuenta las calificaciones obtenidas de los modelos de calidad presentados al aplicarles los criterios de selección, puede observarse que algunos de los modelos tienen características similares en relación a lo que busca este proyecto. Es importante notar que modelos como McCall, Boehm y FURPS, no sólo son los más antiguos sino también han servido como base para la construcción de modelos posteriores, esto es importante de notar pues es posible ver cómo ha sido la construcción de modelos para la valoración del producto software desde que fueron planteados éstos primeros. Para mostrar esta percepción de la evolución de los modelos de calidad del producto software, es mostrada una posible relación de descendencia en Figura 3:

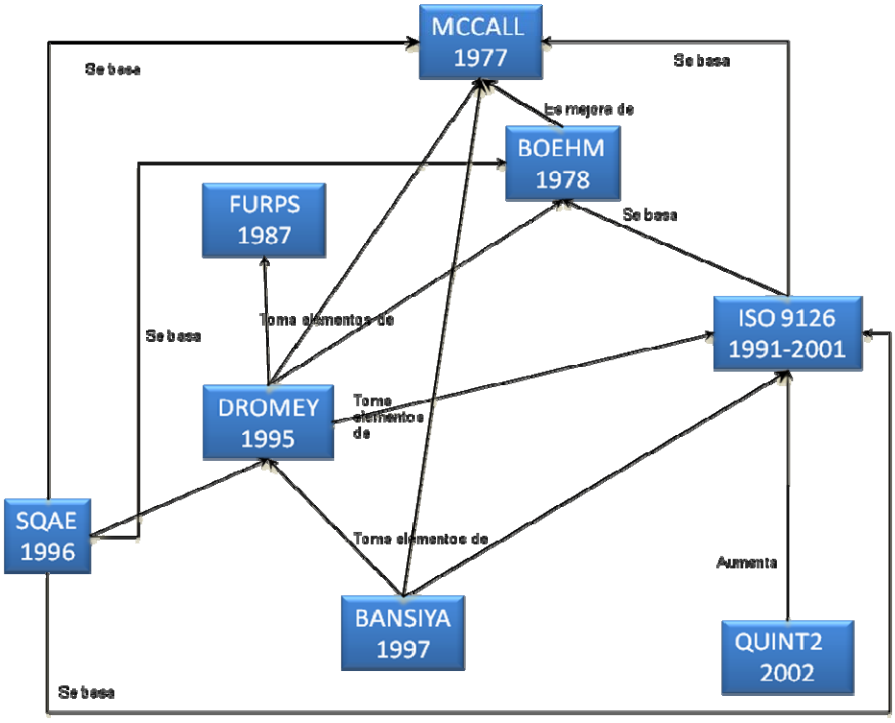

Figura 3. Relación entre modelos

Como puede observarse, los modelos están basados en anteriores para presentar su percepción de la calidad del producto software, y los elementos que componen a cada uno. Por ejemplo en el caso de SQAE, tiene sus orígenes en ISO 9126 en su versión del año 1991, pero fue desarrollado paralelamente con la versión presentada en 2001, mientras que Bansiya por su parte está fundamentado en los modelos de McCall, Dromey e ISO-9126. De otro lado, nótese cómo el modelo FURPS ha sido inspiración de alguna manera para el modelo de Dromey, al igual que otros anteriores, como el modelo de McCall y Boehm. Es por eso que es realizada una selección de aquellos modelos que serían los más básicos y sobre quienes están apoyados unos y otros.

2.1.2. Modelos de calidad para producto software seleccionados.

Teniendo en cuenta la selección hecha hasta aquí, es necesario ampliar la información de los modelos y estándares que servirán como base para el desarrollo de éste proyecto. Los modelos y estándares de calidad mostrados a continuación presentan propuestas o perspectivas frente a cómo valorar la calidad del producto, además de tratarse con cierta profundidad respecto a otros [52][53]

McCall - 1977 [1][18]

El modelo fue escrito por McCall, Richards y Walters, siendo publicado en el año 1977 en un documento denominado “Factors in software quality”. El modelo refleja perspectivas del desarrollador y del usuario, además presenta una estructura jerárquica bidireccional [54] para organizar los factores que están divididos en tres aspectos de calidad de software (revisión, transición y operación), mostrado en la Figura 4.

Los factores de calidad planteados por McCall son medidos a través de 21 criterios o métricas de calidad que él propone; el problema es que dichos criterios son calculados a través de preguntas dicotómicas del tipo “SI”-”NO”, las cuales son contestadas por una o varias personas, esto podría implicar subjetividad dado que cada individuo puede evaluar la calidad de forma diferente.

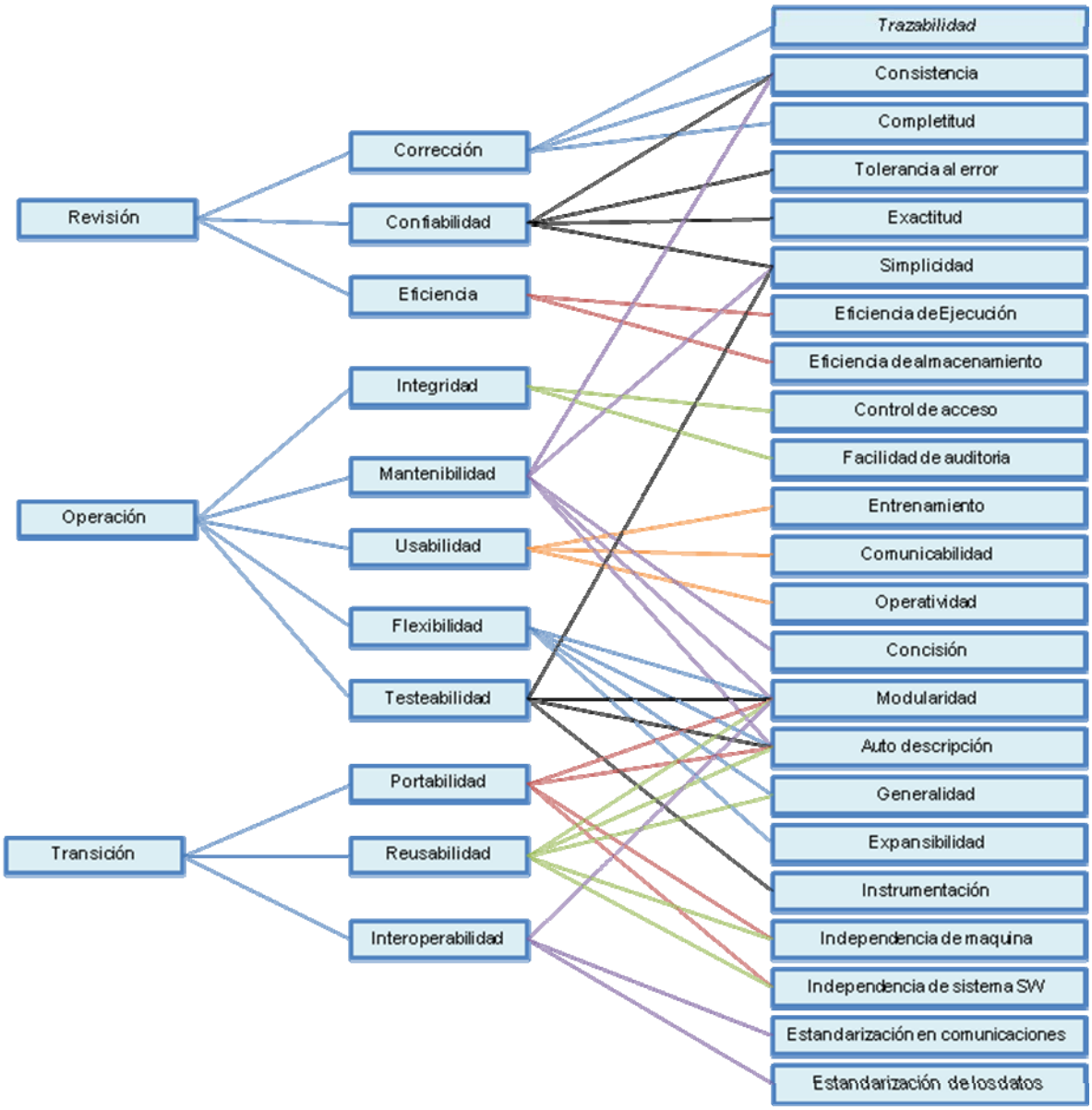

Figura 4. Factores de calidad de McCall

El modelo de McCall ha recibido críticas por considerar que su juicio de medición de la calidad es subjetivo, pues está basado en opiniones personales de quien esté respondiendo las preguntas que el modelo fórmula [3]. El modelo fue desarrollado con el objetivo de mejorar la calidad de los productos software [22]. Su mayor aporte fue el establecimiento de las relaciones entre características de calidad y métricas [8] lo que ha hecho que este modelo sea un predecesor de los modelos actuales.

BOEHM – 1978 [17]

El segundo modelo presentado en este documento es el propuesto por Barry Boehm et al[5] en 1978. Éste define la calidad de software en términos de atributos cualitativos y los mide usando métricas. El modelo no es muy distinto al de McCall, porque muchos de sus factores de calidad son los mismos. Éste modelo también presenta sus factores de calidad estructurados jerárquicamente de alto a bajo nivel como es mostrado a continuación (ver Figura 5):

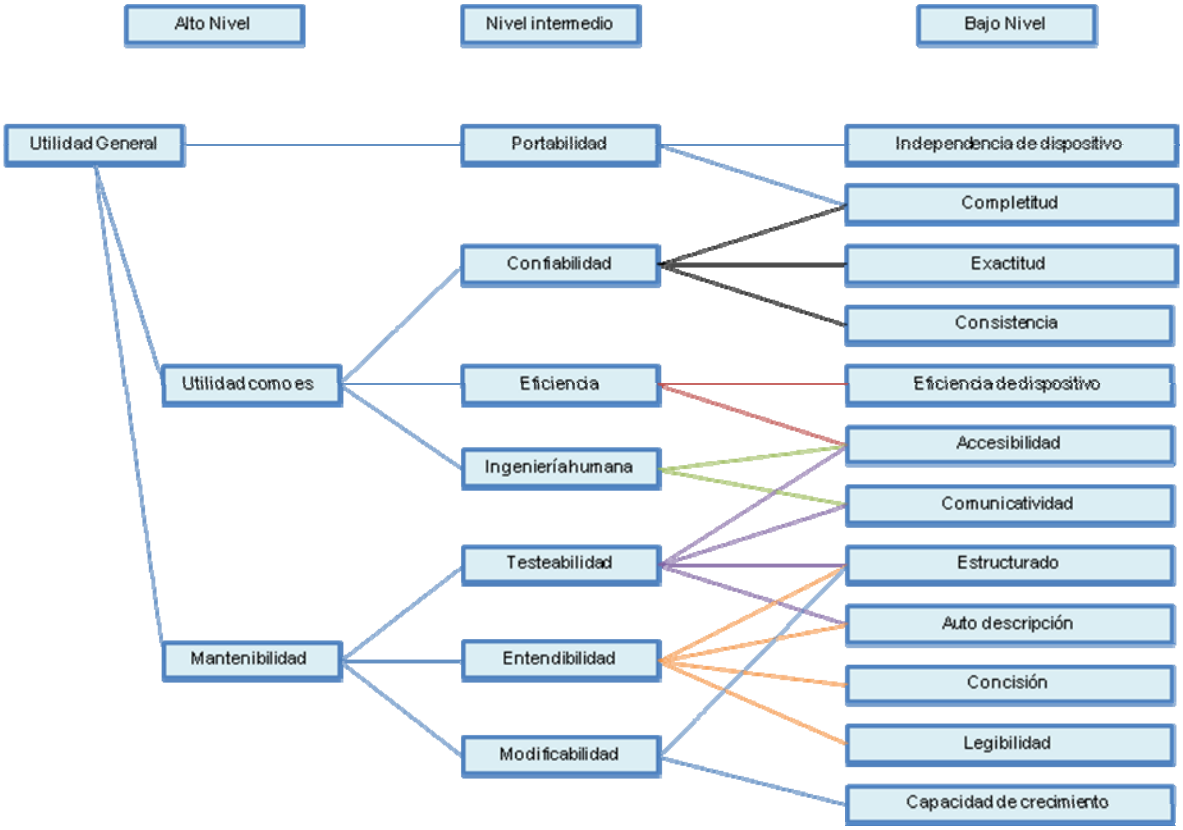

Figura 5. Factores de calidad de Boehm

El modelo de Boehm es un importante predecesor de los modelos más actuales. Básicamente el modelo trata de dar una medida de la calidad de software cuantitativamente a través de sus atributos y métricas [3][35]. El problema con el texto propuesto por Boehm[17] es que presenta las métricas sin fórmula, lo cual puede ser confuso para su aplicación.

[5] Hans Kaspar, Myron Lipow y Michael Merritt.

FURPS – 1987 [20]

En 1987 Hewlett-Packard desarrolló una serie de factores de calidad que reciben el acrónimo de FURPS, que incluye cinco (5) categorías principales por sus nombres en inglés: Funcionalidad (Functionality), Usabilidad (Usability), Confiabilidad (Reliability), Desempeño (Performance) y Soportabilidad (Supportability), de aquí el nombre del modelo (ver Figura 6).

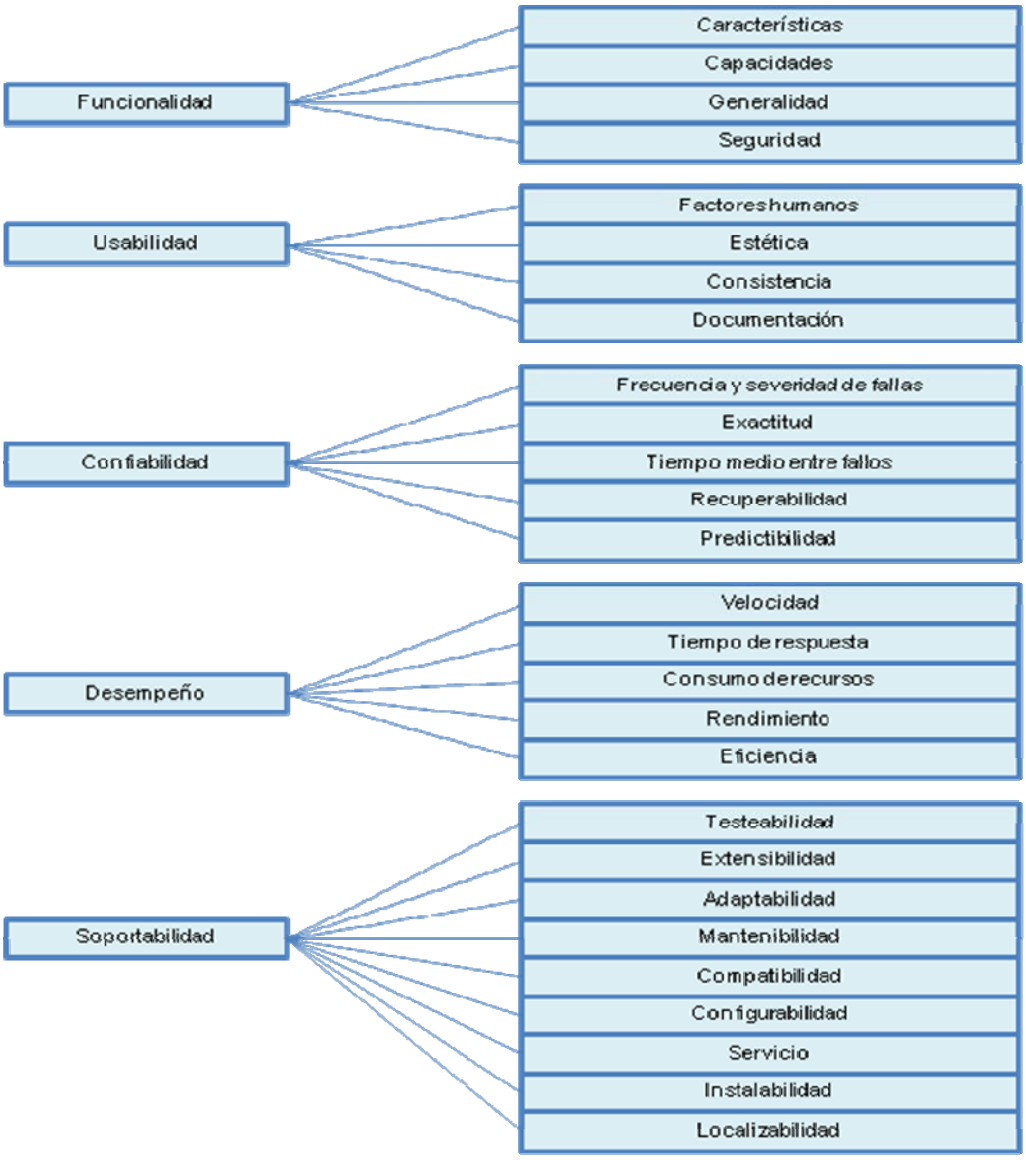

Figura 6. Modelo de FURPS

El modelo es bastante flexible, sin embargo una desventaja que presenta es que no considera la característica de calidad [54]. Otro punto es que en el libro de Grady & Caswell [20], considerado como base del modelo, no hay claridad en elementos como atributos y métricas, para ello al parecer existe otro documento que está restringido al público en general.

ISO 9126 - (1991/2001) [15]

El estándar ISO 9126 presenta su primera versión en 1991, luego en 2001 es remplazado por ISO 9126:1, además el estándar completo cuenta con tres ítems adicionales para ayudar a la mejora de la calidad del producto software (Métricas externas, Métricas internas, Métricas de calidad en uso). Adicionalmente presenta una estrecha relación con el estándar ISO 14598:1. El estándar ISO-9126 define un modelo, basado en modelos ya existentes como McCall, Boehm y US Air Force

El estándar ISO 9126 presenta dos partes, el Modelo de calidad para calidad externa e interna, y el Modelo de calidad para calidad en uso. La primera parte del modelo especifica 6 características de calidad externa, las cuales están divididas en sub-características que representan la calidad interna y tienen influencia sobre las características externas. La segunda presenta cuatro características de calidad, para ser evaluadas desde la vista del usuario (ver Figura 7 y Tabla 4).

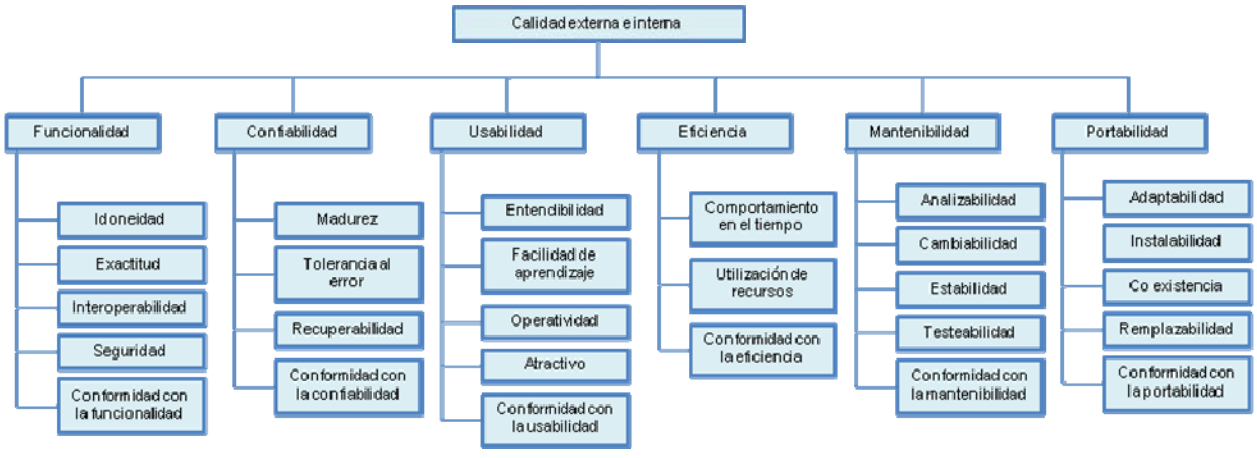

Figura 7. Modelo de calidad para la calidad externa e interna

Calidad en Uso			
Efectividad	Productividad	Seguridad	Satisfacción

Tabla 4. Modelo de calidad para calidad en uso

En su primera versión en 1991, unificó conceptos de calidad de otros modelos, sin embargo no definía atributos y métricas para diferentes dominios y metas de evaluación, faltaba información consistente [13]. Luego, la versión de 2001 implementó un juego de métricas complementando la información faltante de atributos y métricas, sin embargo éstas presentan una forma abstracta, de manera que deben ser aplicadas con algo de cuidado [55].

DROMEY – 1995 [26][54]

El modelo de Dromey presenta un esquema de 6 relaciones binarias entre 3 entidades definidas (Conjunto de componentes, propiedades que acarrean calidad de los componentes, atributos de calidad de alto nivel) en la Figura 8, cuatro de las cuales permiten evaluar la calidad desde la perspectiva del producto o el proceso.

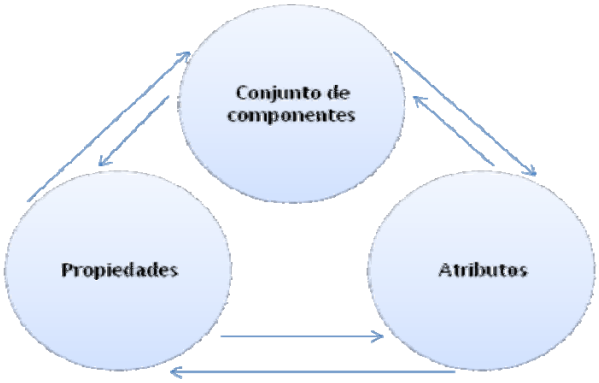

Figura 8. Esquema general del modelo de Dromey

Éste modelo propone una alternativa al inconveniente presentado cuando los atributos de alto nivel no pueden ser medidos directamente sobre el software. En respuesta a esto, los atributos de alto nivel pueden obtenerse con la construcción de componentes que representen un conjunto de propiedades del producto, señalando aquellas que afectan los atributos de calidad.

El siguiente cuadro resume los atributos utilizados en el modelo. (ver Tabla 5)

Atributos						
Funcionalidad	Fiabilidad	Usabilidad	Eficiencia	Mantenibilidad	Portabilidad	Reusabilidad

Tabla 5. Atributos de calidad de Dromey

Una ventaja importante del modelo de Dromey es que puede brindar asistencia en la búsqueda sistemática de defectos de calidad[26].

SQAE (Software Quality Assessment Exercise) – 1995 [29]

Robert A. Martin and Lawrence H. Shafer (MITRE) crearon el SQAE para proveer una serie de herramientas y métodos de evaluación que aporte una medida de calidad de software repetible y consistente, además de asociarle el riesgo. El aseguramiento de calidad que provee SQAE está enfocado en el riesgo relacionado con diferentes áreas de calidad y produce una lista de riesgos conducidos y elementos mitigables que pueden ayudar para hacer elecciones juiciosas cuando sean seleccionados desarrolladores y/o mantenedores de software. SQAE está basado en modelos tales como: Boehm, McCall y Dromey, además del estándar ISO/IEC 9126 (desarrollado paralelamente). Las cuatro áreas de calidad con las que SQAE trabaja son: Mantenibilidad, Evolución, Portabilidad y Consistencia, además presenta siete factores (Independencia, Modularidad, Documentación, Auto descripción, Control anomalía, Diseño simple) para medir la calidad. (ver Figura 9)

Es calificado como una metodología muy rápida y económica, que incorpora criterios rígidos para la evaluación de calidad, limitando la naturaleza subjetiva de algunos criterios de evaluación. Ha sido usada por organismos como el Ministerio de Defensa de los Estados Unidos y Ecole de Technologie Superieure[29][30].

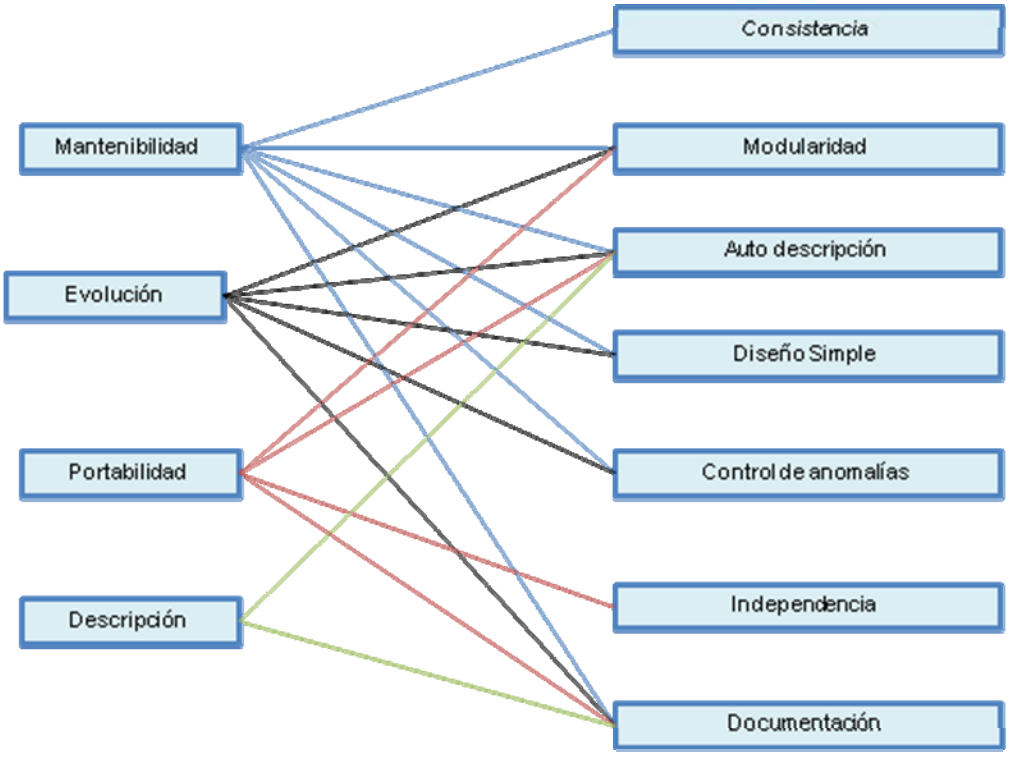

Figura 9. Modelo SQAE

BANSIYA – 1997 [31]

El modelo denominado QMOOD+ (Quality Model for Object-Oriented Design) fue desarrollado por Jagdish Bansiya y Carl G. Davis; basado en otros como McCall, Dromey, e ISO 9126. Este framework es una metodología para el desarrollo de modelos de calidad de estilo Bottom-up (de abajo hacia arriba), proporcionando un acercamiento, el cual asegure que los detalles de bajo nivel sean bien especificados y computables. Consta de cuatro niveles (L1 a L4) conectados entre sí, además contiene 6 atributos de calidad (Reusabilidad, flexibilidad, comprensibilidad, funcionalidad, extensibilidad y efectividad), (Ver Figura 10), y contiene 11 propiedades de diseño que complementan a los atributos, (Ver Figura 11). Además posee una serie de métricas para la evaluación de los mismos.

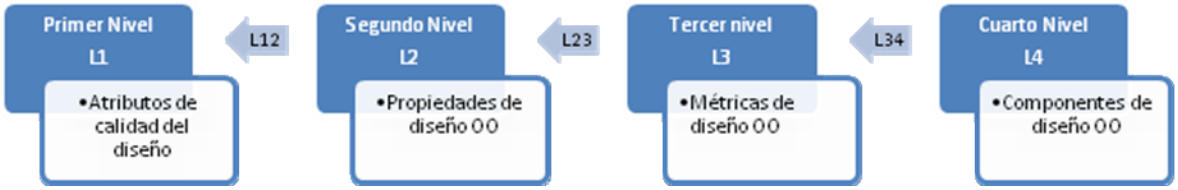

Figura 10. Niveles y enlaces del modelo de Bansiya (QMOOD)

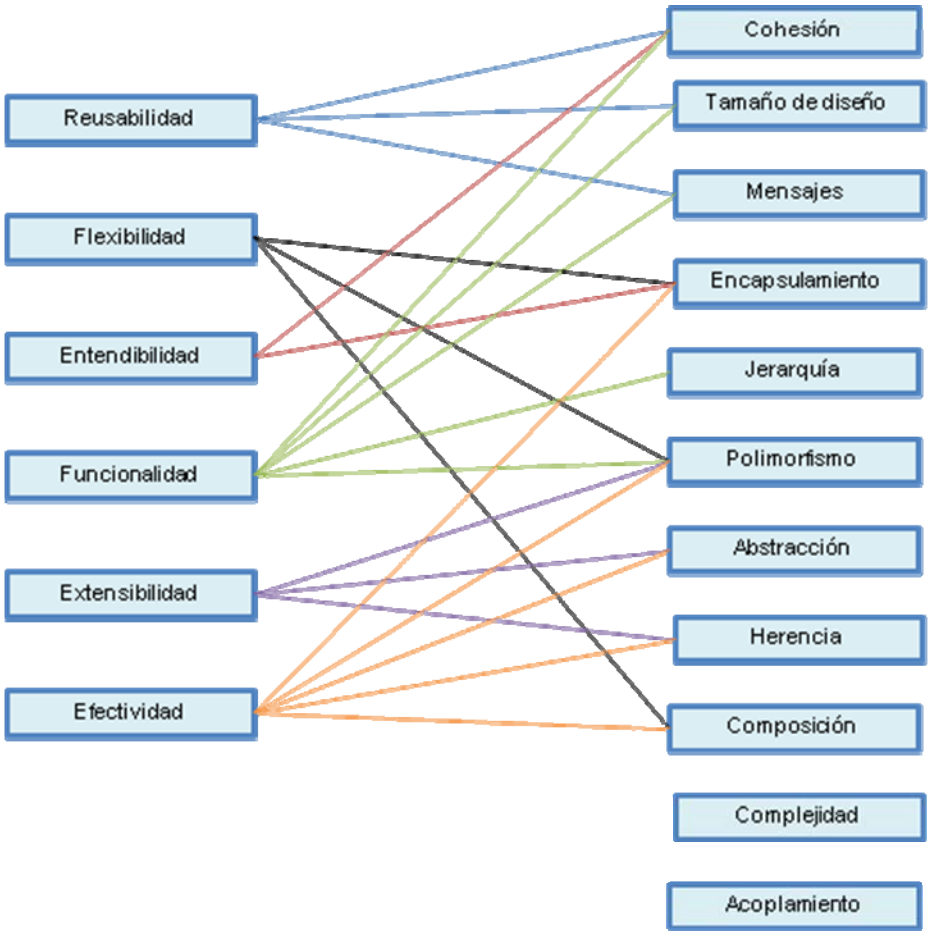

Figura 11. Atributos del Modelo de Bansiya

El modelo de Bansiya utiliza un conjunto de seis atributos, cuatro han sido retomados desde ISO 9126; la forma de calcular estos atributos está hecha de forma intuitiva, asignando pesos positivos o negativos, siendo posible hacer variaciones ajustables a los objetivos de la organización que haga uso del modelo. Un problema con este modelo es que deja de usar algunos elementos de orientación a objetos [53], [57].

QUINT2 -2002 [58][59]

Este modelo es autodenominado “modelo de calidad de software ISO extendido”, dado que es un súper-conjunto del grupo de características y sub-características del modelo ISO 9126. La estructura general del modelo es la siguiente (ver Figura 12):

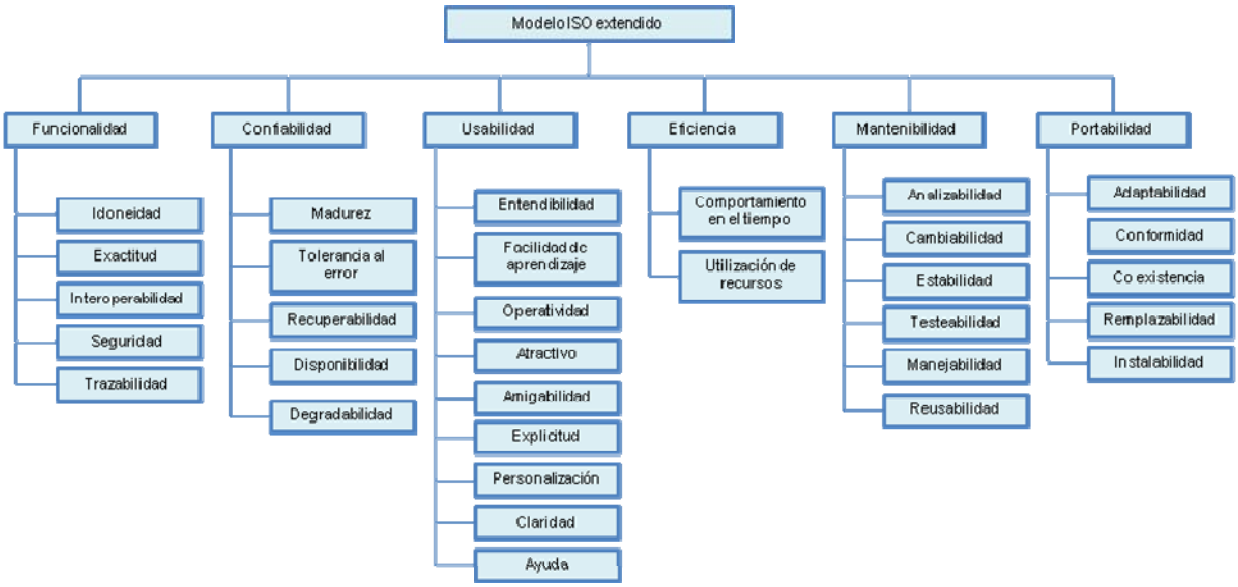

Figura 12. Atributos de Quint2

El modelo de calidad QUINT2 también presenta una ampliación de la norma ISO 9126, pensada para valorar la calidad de arquitecturas software [60]. Aumenta características más apropiadas para valorar productos Web.

En este punto parece evidenciarse, según la bibliografía, que existe un antecedente entre un modelo y otro. Tratando entonces de encontrar la relación que existe entre ellos, a partir de los elementos que los componen; es posible pensar que estos elementos son los atributos de calidad, los cuales son tratados de formas similares en cada modelo. Aún cuando algunos modelos nombran los atributos de manera diferente, en esencia puede tratarse de lo mismo, además de tener en consideración el modo en que pueden ser aplicados los atributos en cada modelo, y como consecuencia evidenciar si conservan sus cualidades, convirtiéndose en elementos básicos desde los que pueden realizarse la medición de calidad de software. Dicho antecedente implicaría una clasificación entre los elementos de un modelo y otro, lo cual intenta realizarse a continuación.

2.2. Esquema de clasificación

Para clasificar los atributos y métricas es planteado el uso de un ***espectro***, el cual nos indica por medio de los puntajes obtenidos después de aplicar los criterios, presentados en los numerales 2.3.1 (criterios de clasificación de atributos) y 2.4.1 (criterio de clasificación de métricas), proponiendo el uso de tres gamas:

- *Claro*: los elementos claros son aquellos que poseen un puntaje alto en el totalizado de los criterios.
- *Gris*: los elementos que cumplen con al menos algunas de las características deseables.
- *Oscuro*: los elementos que no cumplen con los requisitos, o están en un nivel inferior debido a sus propiedades.

Al aplicar el espectro, esto no trata de decir que unos atributos y métricas son mejores que otros, sino más bien que hay algunos elementos que merecen mayor estudio y desarrollo en el futuro. Para comprender mejor el rango puede verse la Figura 13

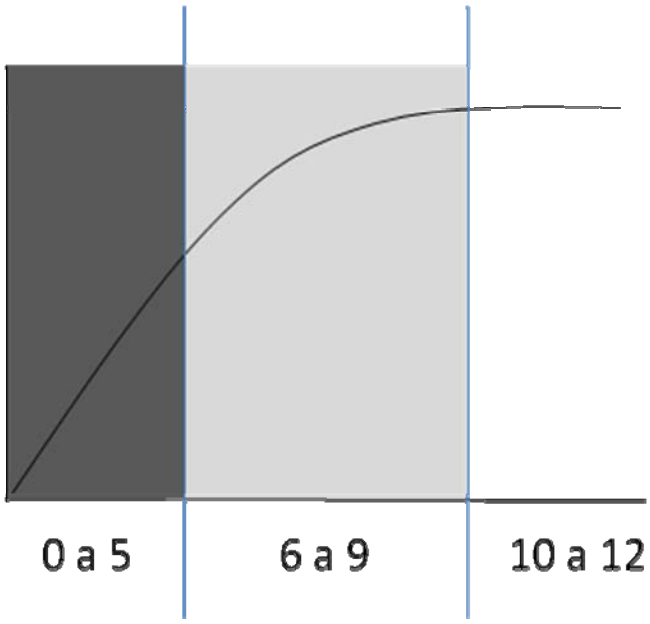

Figura 13. Escala de Clasificación

El espectro pretende realizar una clasificación de los elementos presentes en los diferentes modelos. En las siguientes secciones es mostrado dicho proceso para los atributos (sección 2.3) y las métricas (sección 2.4) de calidad de software.

2.3. Los atributos de calidad para producto software

Hasta aquí, cada modelo presenta un grupo de atributos tanto externos como internos. En este punto interesa conocer en números éste indicador para cada modelo.

	Numero de Atributos		
MODELO/ESTÁNDAR	Internos	Externos	Total
McCall	23	11	34
Boehm	12	8	20
FURPS	27	5	32
Dromey	26	7	33
SQAE	7	4	11
Bansiya	11	6	17
ISO 9126	27	6	33
QUINT2	31	6	37

Tabla 6. Número de atributos por modelo

Como es posible observar en la Tabla 6, cada modelo posee un número pequeño de atributos externos, y un número mayor de atributos internos. Además de esto es posible comparar los modelos entre sí para relacionar con qué frecuencia aparece un atributo (Ver anexo B), es así por ejemplo que el atributo más mencionado por los modelos es la *Mantenibilidad* apareciendo en 7 de los 8 modelos que son estudiados aquí, esto podría sugerir que éste atributo es considerado uno de los más importantes; por otro lado, existen diversos atributos que tan sólo son mencionados en un modelo como es el caso de *Ingeniería humana* que solo aparece en Boehm, esto no quiere decir que carezca de importancia, sino que puede tratarse de un cambio de nombre en modelos posteriores, o en el peor de los casos es un atributo en desuso. El capítulo 3 tratará los atributos encontrados en los modelos.

2.3.1. Criterios de clasificación para los atributos de calidad

Es importante clasificar los atributos, optando por hacer uso de criterios, los cuales están basados en parte de otros trabajos [61] y por otra parte de las definiciones que los mismos atributos dan, adaptadas a las necesidades del proyecto; es así como son propuestos los siguientes criterios:

- A1: *Relación con métricas*: indica si el atributo tiene métricas relacionadas, puede ser por medio directo o indirecto.
  1: No fueron encontradas métricas
  2: Fueron encontradas algunas métricas pero están incompletas o no son claras.
  3: Tiene métricas asociadas claras.

- A2: *Completitud*: es referida a la información encontrada del atributo.
  1: No fue encontrada información, es casi nula.
  2: La información encontrada es limitada e incompleta.
  3: El atributo ha sido suficientemente documentado. Es mencionado por diversos autores.

- A3: *Simplicidad*: Grado en que el atributo es claro y fácil de entender.
  1: El atributo no tiene información que sustente su comprensión.
  2: El atributo posee información pero es ambigua, puede tener diferentes interpretaciones.
  3: El atributo es claro y fácil de entender.

Después de revisar los modelos, aparecen cerca de 134 atributos (ver anexo C), a los que les fueron aplicados los criterios de clasificación, obteniendo que 67 pertenecen al espectro oscuro, 12 al gris y 55 al claro. La organización de los atributos puede ser vista en el Capitulo 3.

2.4. Las métricas de calidad para producto software

Para el caso de las métricas es complicado encontrarlas todas en un solo modelo. Algunos como ISO 9126 presentan métricas bien definidas para calcular la calidad interna y externa, brindando una amplia gama de mediciones, que pueden ser llevadas a cálculos matemáticos, sin embargo modelos como McCall presentan más subjetividad, aunque el modelo brinda una serie de formulas para el cálculo de la calidad a través de métricas, éstas en su mayoría, son calculadas con datos que brinda el evaluador de calidad, aumentando la incertidumbre en los resultados, debido a esto no es claro cómo debe usarse la escala, ni cuándo.

En los documentos dedicados al estudio de la medición de calidad puede encontrarse un sin número de métricas para cada etapa del software, algunas muy bien documentadas y otras no tanto, por esta razón, este trabajo ve la necesidad de clasificarlas y ubicarlas en el espectro.

2.4.1. Criterios de clasificación de las métricas de calidad

Hasta el momento han sido encontradas alrededor de 617 métricas relacionadas a los atributos presentados por los modelos mencionados anteriormente, sin embargo no todas son fáciles de calcular, o son usables para cualquier tipo de software. Para ello son planteados los siguientes criterios [61]:

- Mt1: *Completitud*: Referido a la bibliografía, la cantidad de documentos encontrados donde aparece la métrica, y qué tan documentada esta.

  1: No existe información sobre la métrica o no es suficiente. La mención por parte de los autores es casi nula.

  2: Existe información pero está incompleta, es posible hacer interpretaciones sobre la métrica. La mención hecha por autores es limitada.

  3: la métrica está bien documentada. Es mencionada por diversos autores.

- Mt2: *Computable*: Indica si es posible llevar la métrica a términos matemáticos para posteriormente ser puesta en funcionamiento a través de un programa computacional.

  1: La métrica es totalmente subjetiva; no es posible computarizarla

  2: Es posible computarizarla. Tal vez necesita intervención humana para ser calculada.

  3: Es posible computarizar la métrica, está en términos matemáticos claros.

- Mt3: *Objetividad*: Grado en que la métrica evita interpretaciones individuales. En cualquier opinión los valores han de ser idénticos

  1 y 2: La métrica puede tener interpretaciones diferentes.

  3: La métrica no tiene interpretaciones diferentes.

- Mt4: *Simple*: Grado en que la métrica es fácil entender por el lector. Tanto su definición como su uso es simple. Una métrica simple no puede ser ambigua.

  1: La métrica es compleja o su interpretación es ambigua.

  2: La métrica es compleja pero puede ser entendida con algo de estudio.

  3: La métrica es fácil de entender y usar.

Estos criterios indican una serie de requisitos que las métricas deben cumplir, sin embargo algunas métricas no están dentro de estos parámetros. En este caso, el criterio no aplica (NA) y no otorga ningún valor al criterio.

Después de la aplicación de los criterios a las métricas (ver Anexo D), éstas fueron clasificadas de la siguiente manera: 19 métricas en el espectro claro, 334 en el gris y 264 en el oscuro.

2.5. Las heurísticas de calidad para producto software

Un proceso similar al aplicado a métricas, es realizado a las heurísticas de calidad para el producto software; con la diferencia que la gran mayoría de ellas están recopiladas en la obra de Riel [82], la cual contiene una vasta clasificación. Sin embargo, diversos trabajos han sido realizados sobre la misma obra, donde obtienen diferentes puntos de vista

respecto a la clasificación ahí presentada, a causa de la poca y clara documentación, que ha llevado a múltiples interpretaciones.

2.5.1. Criterios de clasificación de las heurísticas de calidad

Para este ítem la forma de seleccionar es algo más flexible que la presentada anteriormente para la clasificación de métricas; por esta razón están planteados ciertos criterios considerados deseables en una heurística, de tal forma que a mayor cantidad de criterios cumplidos para cada una, mayor opción tendrá en hacer parte del presente compendio. Así entonces, los criterios son:

- Ht1: *Frecuencia de aparición*: La heurística ha sido tratada por más de un autor.
- Ht2: *Es simple*: La heurística de diseño está escrita en lenguaje simple, claro y entendible de manera que es fácil de usar.
- Ht3: *Exacto*: La heurística está escrita de manera que sea precisa y dirigida hacia un aspecto específico del diseño de software.
- Ht4: *Contradicciones*: La aplicación de la heurística de diseño no contradice la aplicación de otra, salvo si pertenecen a un mismo grupo de heurísticas
- Ht5: *Interpretaciones*: La heurística, en su definición, presenta una y solo una interpretación como recomendación para el diseño de software(es decir, cuál es la recomendación y dónde debe (ría) ser factible aplicarla).

Cada uno de los anteriores es aplicado a una heurística dada, de la siguiente manera:

1: La heurística no satisface el criterio planteado.

2: La heurística satisface el criterio planteado.

Al aplicarlo a cada heurística, es obtenido un puntaje total, siendo 10 el máximo posible.

Para que una heurística en particular pueda hacer parte del conjunto final de heurísticas, determine el puntaje total obtenido, y si éste es igual o superior a 8, la heurística puede ingresar al mencionado conjunto.

El proceso de clasificación de las heurísticas está en el Anexo E, donde muestra las principales recopilaciones realizadas por diversos autores. Algunas de las heurísticas

seleccionadas, de acuerdo a los criterios anteriormente mencionados, están en el capítulo 3.

2.6. Conclusión del capítulo

Los elementos antes mostrados que hacen parte de atributos, métricas y heurísticas de calidad de software, permiten resaltar que la presencia de los primeros en un software marca una pauta que deja entrever posibles falencias en el diseño de los productos software. De manera similar, la valoración de calidad de software a partir de métricas de calidad de software también permite resaltar la aparición de falencias, de manera que éstas pueden obtenerse a través de cálculos matemáticos. Tanto los atributos como las métricas permiten determinar el grado presentado por las características de calidad en un producto software.

Por otra parte, las heurísticas de calidad de software ofrecen, a los diseños y diseñadores, un medio que orienta a mejorar la calidad de los productos software.

Así pues, los atributos, métricas y heurísticas de calidad de software están estrechamente ligados, tal que permiten a las personas interesadas en la calidad de software, gozar de un mecanismo para evaluarla en un producto dado, ofreciendo una vasta fuente de información propia para el tipo de evaluación que espera mostrarse en los capítulos siguientes.

CAPÍTULO III: MARCO CONCEPTUAL PARA LA VALORACIÓN DEL PRODUCTO SOFTWARE ORIENTADO A OBJETOS

Este capítulo plasma el marco conceptual propuesto para exponer los elementos de calidad encontrados en el estudio realizado sobre modelos, atributos, métricas y heurísticas de calidad para el producto software. El capítulo está dividido en cuatro partes, la primera muestra una estructura propuesta en este trabajo, la segunda expone la organización de los atributos de calidad encontrados y sus definiciones, la tercera parte muestra las métricas de calidad, y las heurísticas en la cuarta parte. Éste capítulo es la contribución principal al cumplimiento del objetivo general, y el primero de los objetivos específicos, de conformación de un compendio de atributos y métricas de calidad a partir de la revisión exhaustiva de modelos y estándares de calidad, los cuales fueron presentados en el capítulo 2; un aporte que contribuye a la conformación del marco es la presentación de las relaciones de atributo con métricas o heurísticas de calidad en un solo documento pues estos elementos en general estaban por separado.

3.1. Organización propuesta para el marco conceptual.

En los modelos de calidad estudiados ha sido visto el uso de estructuras jerárquicas para sus atributos, sub-atributos y métricas, y alguna clasificación de heurísticas; éste trabajo trata de incluir los cuatro elementos (atributos, sub-atributos, métricas y heurísticas de calidad), y debido a que los modelos y estándares no siempre son similares a la hora de clasificar cada elemento en la jerarquía, éste proyecto propone una organización diferente, sin dejar de lado los trabajos anteriores, teniendo en cuenta que ésta organización es una refinación de un trabajo previo, expuesto en el artículo realizado por Moreno [6] llamado “Compilación de un Modelo para Evaluar Atributos de Calidad en Productos Software”, completado con la información reunida durante la elaboración del presente trabajo. En aquél trabajo, hay un primer acercamiento a los modelos de calidad y los atributos, dando la organización inicial en capas. El actual pretende mostrar una estructura de los atributos, sub-atributos, métricas y heurísticas, que los modelos aquí mencionados poseen. Es presentado en capas para su mejor comprensión:

Capa 1. ***Atributos de calidad de alto nivel***: Esta capa organiza los factores de calidad que son independientes entre sí. El atributo es considerado como externo. Representan atributos abstractos.

Capa 2. ***Atributos de calidad Intermedios***: Esta capa indica todos los factores de calidad cuya valoración contribuye a la valoración de los factores de alto nivel que están en la capa 1; a su vez, estos son atributos externos. Pueden ser compartidos o exclusivos.

Capa 3. ***Atributos de calidad de bajo nivel***: Aquí están los atributos internos, que tienen influencia sobre los externos. Pueden ser compartidos o exclusivos.

Los atributos de las capas 2 y 3 poseen una cualidad de ser compartidos o exclusivos. Un atributos *Compartido* es aquel que le sirve a más de un atributo externo y uno *Exclusivo es* aquel que sólo le sirve a un atributo externo o interno únicamente. Como la clasificación de atributos no es estrictamente jerárquica, algunos de los elementos de capa 3 pueden contribuir al cálculo de los atributos de capa 1 y 2.

Capa 4. ***Métricas***: Esta capa posee las métricas que soportan la valoración de los atributos de las capas anteriores.

Capa 5. ***Heurísticas***: Esta capa está compuesta por las heurísticas encontradas en diversos trabajos.

La anterior organización puede apreciarse gráficamente como muestra la Figura 14:

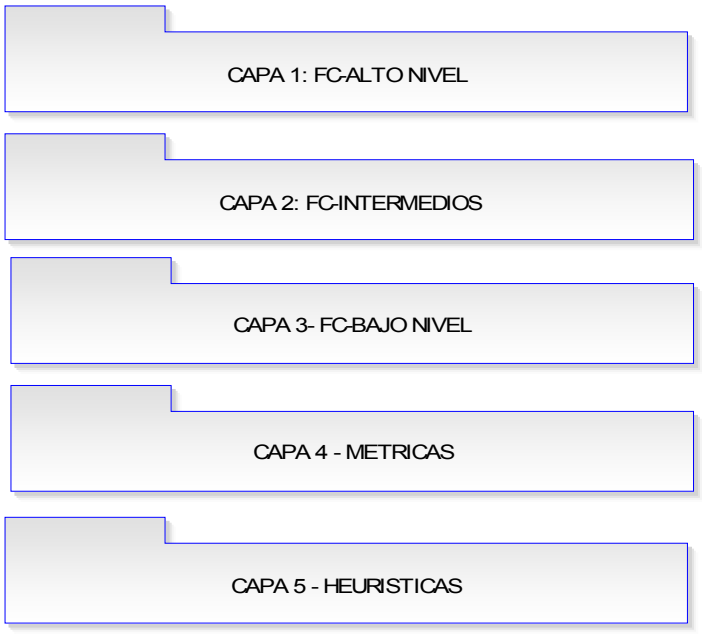

Figura 14. Capas del marco conceptual

Las capas ayudan a tener una visión diferente de la organización de los atributos, métricas y heurísticas, en donde no necesariamente pierde la jerarquía planteada por los modelos de calidad para producto, sino más bien representa una forma de clasificar los elementos para su más fácil manejo. En las próximas secciones (3.2 a 3.4) es ampliada cada una de las capas antes mencionadas.

3.2. Los atributos de calidad

En el capítulo 2, sección 2.3, en relación a los modelos de calidad de software, puede percibirse que cada autor maneja una notación diferente para referirse a un mismo concepto. McCall [18] utiliza factores y criterios; mientras que Boehm [17] utiliza características y primitivas; para FURPS [20] son factores y atributos; de otro modo, ISO 9126 [15] maneja atributos o características y sub-atributos de calidad, y para Dromey [26] son atributos y propiedades de calidad de software. Todos estos conceptos hacen referencia a clasificaciones que cada autor hace para su modelo, en definitiva indican cómo va a estar dada la medida de calidad. Para llegar a una clasificación ordenada de dichos conceptos, apoyándonos en Meyer [62], llegamos a la siguiente unificación para los términos aquí utilizados, en la Tabla 7:

	McCall	Boehm	FURPS	ISO 9126	Dromey	Bansiya
Concepto	Factor	Característica Primitiva	Factor	Atributo ó Característica	Atributo	Atributo de calidad
	Criterio	Característica	Atributo	Sub - Atributo	Propiedad	Propiedad de diseño
	Métrica	Métrica	-	Métrica	Métrica	Métrica

Tabla 7. Selección de conceptos por modelo

En la tabla previa puede apreciarse como las notaciones de las palabras para cada modelo hacen referencia a un mismo concepto; en la primera línea hay diferentes definiciones para lo que al final trata de llamarse factor externo, los cuales pueden ser percibidos por los usuarios de un software, y pueden ser medidos por los factores internos (presentados en la segunda fila de los conceptos), que no son percibidos por el usuario sino por personas en las organizaciones encargadas del diseño e implementación. Algo en lo que al parecer los autores concuerdan es en denominar métricas a la forma de calcularlas o hacer la medida cuantitativa para satisfacer el cumplimiento del factor interno [62]. Para este trabajo hemos decidido tomar la conceptualización de Atributo (Factor externo), Sub-Atributo (Factor interno) y métrica, entendido que el atributo es aquél encontrado en un nivel más alto y por tanto su medida es externa, el sub-atributo está en un nivel más bajo y su medida es interna y la métrica es tomada como la medida del sub-atributo.

La naturaleza de un atributo está determinada por la caracterización que busca hacer en el software a evaluar, esto es, si un atributo puede ser evaluado “a simple vista” sin necesidad de ingresar al código fuente con miras a “mejorar” la característica a observar. Así, atributos tales como la usabilidad y la eficiencia pueden ser evaluados a través de criterios que pueden ser expresados con solamente el análisis visual de su desempeño. Por lo tanto, puede determinarse que los atributos pueden ser de naturaleza externa ó interna según la característica software a evaluar.

Este capítulo procura mostrar una comparación entre atributos de los modelos de calidad del producto software en relación a los atributos con el fin de seleccionar aquellos que tenga mayor coincidencia, importancia o influencia sobre otros atributos de calidad (la totalidad de atributos encontrados puede verse en el anexo C de atributos del capítulo 2).

Por lo tanto, mostramos a continuación la Tabla 8 con los modelos elegidos previamente desde los cuales los atributos, con mayor coincidencia en aparición dentro de los modelos, son extraídos:

(Convenciones utilizadas en la Tabla 8. M1: McCall, M2: Boehm, M3: FURPS, M4: Dromey, M5: SQAE, M6: Bansiya, M7: ISO 9126, M8: Quint 2)

MODELO/ESTANDAR vs. ATRIBUTOS	M1	M2	M3	M4	M5	M6	M7	M8	Frecuencia atributo
Mantenibilidad	x	x	x	x	x		x	x	7
Testeabilidad	x	x	x				x	x	5
Portabilidad	x	x		x	x		x	x	6
Reusabilidad	x			x		x		x	4
Exactitud	x	x	x				x	x	5
Confiabilidad	x	x	x	x			x	x	6
Eficiencia	x	x	x	x			x	x	6
Usabilidad	x		x	x			x	x	5
Comprensibilidad		x				x	x	x	4
Consistencia	x	x	x	x	x				5
Auto descripción	x	x		x	x				4
Funcionalidad			x	x		x	x	x	5

Tabla 8. Modelos vs. Atributos

Los atributos mostrados en la Tabla 8, inicialmente han sido escogidos, debido a su mayor coincidencia de aparición en los diferentes modelos aquí comentados, lo cual, expresa la posibilidad de que exista consistencia en la forma de medir presentada por los modelos de producto, además puede ser un indicio de la influencia que aparenta tener los atributos de bajo nivel sobre los atributos de alto nivel; por otro lado, es considerada su definición en los distintos modelos vistos, que en algunos casos tiende a ser similar. Un aspecto a considerar es que no siempre un atributo posee el mismo nombre en cada uno de los modelos, para esta situación intenta obtenerse la definición más clara, y tomar el nombre más común.

Para poder expresar los atributos encontrados en los modelos, un proceso de clasificación fue aplicado, buscando la relevancia del atributo, por medio de los criterios mencionados en el capítulo 2; es así como logran clasificarse los atributos en una organización que puede verse en la Figura 15.

Es importante resaltar que algunos atributos localizados en el Anexo G no fueron incluidos en la Figura 15, considerando que estaban bajo otro nombre ya incluido.

Como puede observarse, la clasificación indica que el atributo esta en un nivel más alto según su color (espectro), el *claro* (color verde) son aquellos atributos cuya información está más completa, tienen métricas asociadas y son más sencillos de entender; el *gris* (color naranja) indica un nivel intermedio, es decir en alguno de los criterios no cumplió a cabalidad, el *oscuro* (color rojizo) indica que el atributo está incompleto, esto dificulta su estudio.

Cada atributo cuenta con las definiciones de los diversos autores que los citan, para cada atributo es mencionada la sección de descripción mostrada en la definición dada por algunos autores del atributo y una sub sección llamada otras definiciones, donde muestra las percepciones que tienen otros autores acerca del atributo; esta división está hecha pensando en tener una visión más amplia de los atributos presentados y es colocada en la parte de descripción aquella definición que ha sido más trabajada o difundida en los documentos estudiados, además de ser más entendible y madura. Éste documento retoma esas definiciones, expresando lo esencial de cada atributo. En el anexo G puede verse la totalidad de los atributos encontrados, el actual presenta una muestra de los más representativos.

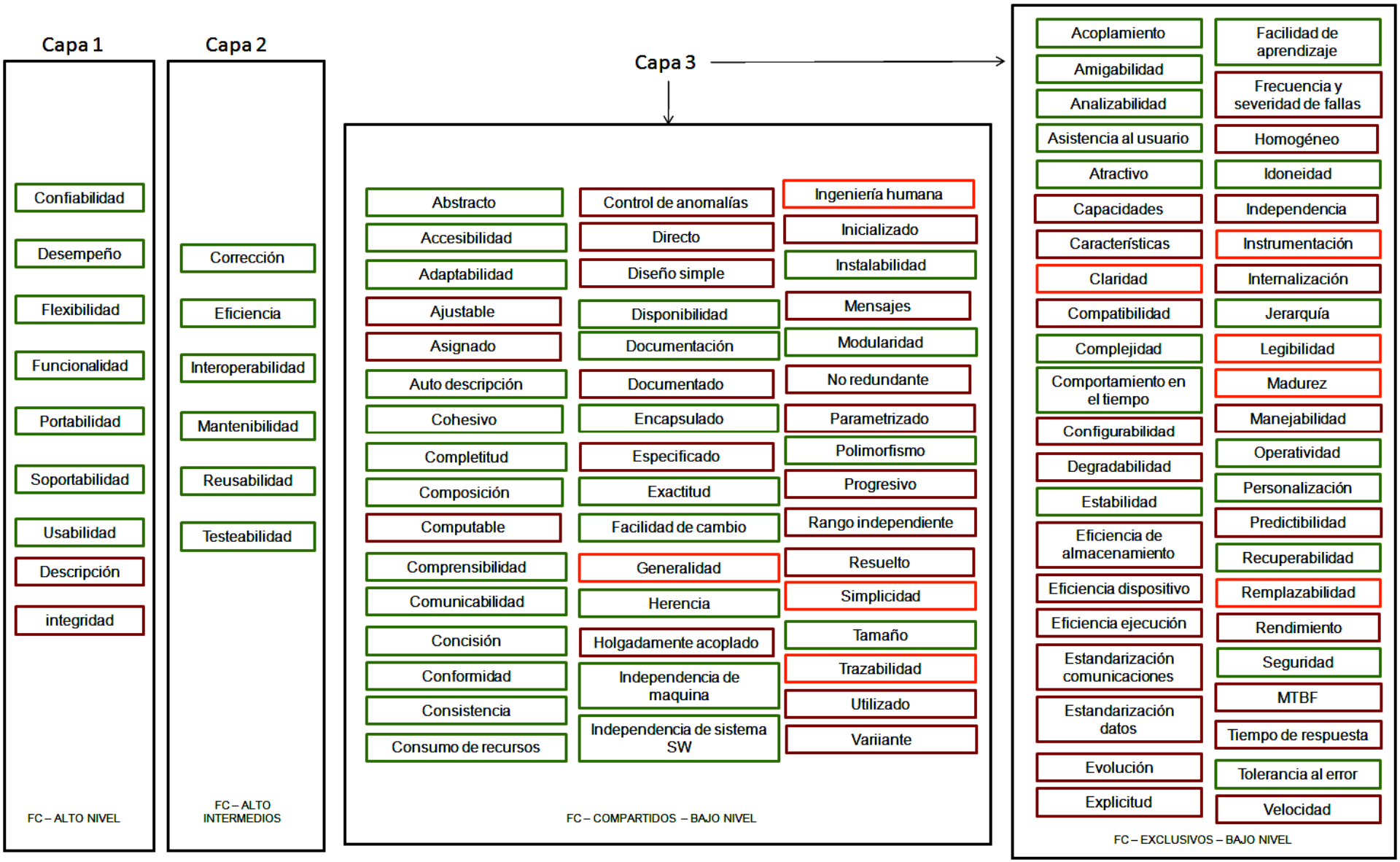

Figura 15. Atributos por capa, basado en “Compilación de un Modelo para Evaluar Atributos de Calidad en Productos Software” [6]

CAPA 1: FC-ALTO NIVEL

CONFIABILIDAD

Nombre en inglés: Reliability

También conocido como: Fiabilidad

Descripción:

McCall [18]: Referido a las funciones con la precisión requerida. ¿Lo hace acertadamente (de forma fiable) todo el tiempo?

Boehm [17]: medida en la que el producto software puede ser el esperado para desempeñar sus funciones pretendidas satisfactoriamente.

Otras definiciones:

FURPS [20]: El MTBF del hardware debe ser al menos cinco veces mejor que el sistema de adquisición de datos

ISO 9126 [15]: La capacidad del producto software para mantener su nivel de desempeño cuando es utilizado bajo condiciones especificadas (durante un determinado período de tiempo).

Quint2 [59]: Capacidad del software para mantener su nivel de desempeño bajo condiciones indicadas por un periodo indicado de tiempo

Gaffney [63]: Medida de un programa puede ser esperado a realizar su función deseada con la precisión requerida.

Alvaro[64]: La característica expresa la habilidad del componente de mantener un nivel especificado de desempeño, cuando es usado bajo condiciones específicas.

Software Quality Assurance Group [65]: Medida mediante la cual de un programa puede esperarse que lleve a cabo sus funciones deseadas sin fallas por un periodo de tiempo dado.

Jetter [3]: Habilidad del sistema para no fallar (McCall)

Pressman[1]: Hasta dónde puede esperarse que un programa lleve a cabo su función con la exactitud requerida. Hay que hacer notar que existen otras definiciones de fiabilidad más completas

Olmedilla [53]: la capacidad del software de mantener un nivel específico de rendimiento bajo determinadas condiciones de uso.

IEEE610.12 [23]: La habilidad de un sistema o componente de realizar sus funciones requeridas bajo condiciones establecidas en un periodo especificado de tiempo.

Atributos relacionados: Consistencia, Tolerancia al error, Exactitud, Simplicidad, Completitud, Madurez, Recuperabilidad, Disponibilidad, Degradabilidad, conformidad, Frecuencia y severidad de fallas, tiempo medio entre fallas, predictibilidad, encapsulamiento, composición, polimorfismo.

Modelos en los que aparece: McCall, Boehm, FURPS, ISO 9126, Dromey, Quint2

USABILIDAD

Nombre en inglés: Usability

También conocido como: facilidad de uso, As utility (Boehm)

Descripción:

McCall [18]: Esfuerzo necesario para aprender, operar, preparar la entrada, e interpretar la salida de un programa. ¿Puedo usarlo?

Boehm[17]: Medida en la que un producto es conveniente y practicable para usar. La interfaz humana es importante aún si el producto no tiene utilidad general

Otras definiciones:

FURPS[20]: No debería ser necesario cambios para programas de aplicación de usuarios

ISO 9126[15]: La capacidad del producto software para ser comprendido, aprendido, utilizado y que sea atractivo para el usuario, cuando es utilizado bajo condiciones especificadas.

Quint2 [59]: Esfuerzo necesitado por uso, y estimación individual de tal uso, por un conjunto de usuarios indicado o implícito

Gaffney [63], Software Quality Assurance Group [65], Pressman[1]: Esfuerzo requerido para aprender a operar, preparar entradas, e interpretar salidas del programa.

Alvaro[64], Olmedilla [53]: Esta característica expresa la habilidad de un componente para ser entendido, aprendido, usado, configurado, y ejecutado, cuando es usado bajo condiciones específicas.

IEEE 610.12 [23]: La facilidad con la cual un usuario puede aprender para operar, preparar entradas para e interpretar salidas de un sistema o componente.

McConnell [66]: La facilidad con la cual los usuarios pueden aprender y usar un programa.

Atributos relacionados: Entrenamiento, comunicatividad, operatividad, Factores humanos, estética, consistencia, documentación, Confiabilidad, eficiencia, ingeniería humana, comprensibilidad, facilidad de aprendizaje, atractivo, conformidad, amigabilidad, explicitud, personalización, claridad, asistencia de usuario

Modelos en los que aparece: McCall, Boehm, FURPS, ISO 9126, Dromey, Quint2

CAPA 2: FC-INTERMEDIOS

EFICIENCIA

Nombre en inglés: Efficiency

También conocido como: Efectividad (Bansiya), eficacia, efectivo (Dromey)

Descripción:

McCall [18]: La cantidad de recursos informáticos y de código requeridos por un programa para realizar una función. ¿Funcionará en mi hardware tan bien como pueda?

Boehm [17]: Medida en la que un producto software cumple a cabalidad su propósito sin malgastar recursos

Otras definiciones:

ISO9126 [15]: La capacidad del producto software para proporcionar un desempeño apropiado, en relación con la cantidad de recurso utilizado, bajo condiciones establecidas

Bansiya [31]: Este es referido a la habilidad del diseño para conseguir la funcionalidad y comportamiento deseados usando conceptos de diseño Orientado a Objetos y técnicas.

Quint2 [59]: Relación entre el nivel de desempeño del software y la cantidad de recursos usados, bajo condiciones indicadas

Gaffney [63], Software Quality Assurance Group [65], Pressman[1]: La cantidad de recursos computacionales y de código requeridos por el software para realizar una función.

Alvaro[64]: La característica expresa la habilidad de un componente para proveer un desempeño apropiado, relativo a la cantidad de recursos usados.

Olmedilla [53]: La capacidad del software de ofrecer el rendimiento apropiado con respecto a la cantidad de recursos utilizados, bajo condiciones prefijadas.

IEEE 610.12[23]: El grado en el que un sistema o componente realiza sus funciones designadas, con un mínimo consumo de recursos.

McConnell [66]: Mínimo uso de los recursos del sistema, incluyendo memoria y tiempo de ejecución.

Atributos relacionados: Eficiencia de ejecución, eficiencia de almacenamiento, eficiencia de dispositivo, accesibilidad, desempeño, comportamiento en el tiempo, utilización de recursos, conformidad, abstracción, encapsulamiento, composición, herencia, polimorfismo

Modelos en los que aparece: McCall, Boehm, FURPS, ISO 9126, Dromey, Bansiya, Quint2

REUSABILIDAD

Nombre en inglés: Reusability

También conocido como: Facilidad de reutilización

Descripción:

McCall [18] : Medida en que un programa puede ser utilizado en otras aplicaciones – relacionado a empaquetamiento y el alcance de las funciones que realizan los programas. ¿Seré capaz de reutilizar algo del software?

Boehm [17]: Refleja la presencia de características de diseño Orientado a objetos, que permite a un diseño ser re-aplicado en un nuevo problema sin esfuerzo significativo

Otras definiciones:

Quint2 [59]: Potencial para ser re-usado completa o parcialmente en otros productos software

Berander [35], IEEE 610.12 [23]: El grado en el que un modulo software u otro producto de trabajo puede ser usado en más de un programa de computo o sistema software.

Gaffney [63]: Medida en que un programa puede ser usado en otras aplicaciones reaccionadas a empaquetamiento y alcance de las funciones que los programas realizan.

Software Quality Assurance Group [65]: Medida en que un software puede ser usado en otras aplicaciones.

Meyer [62], McConnell [66]: Habilidad de elementos software para servir en la construcción de muchas diferentes aplicaciones.

Pressman[1]: (capacidad de reutilización). Hasta dónde puede volverse a emplear un programa *(o* partes de un programa) en otras aplicaciones, en relación al empaquetamiento y alcance de las funciones que realiza el programa

Atributos relacionados: Generalidad, modularidad, independencia de sistema software, independencia de máquina, auto descripción, Tamaño de diseño, Cohesión, Mensajes, Mantenibilidad

Modelos en los que aparece: McCall, Dromey, Bansiya, Quint2

CAPA 3: FC-COMPARTIDOS-BAJO NIVEL

COMPLETITUD

Nombre en inglés: Completeness

También conocido como: compleción.

Descripción:

McCall [18]: Los atributos del software que proporcionan una total implementación de las funciones requeridas

Boehm [17]: Medida en la que todas las partes un producto están presentes, y plenamente desarrolladas.

Otras definiciones:

Dromey [26]: Cuando una forma estructural tiene todos los elementos necesarios para definir e implementar la forma estructural así que esta pueda cumplir a cabalidad sus roles pretendido en una manera que no impacte la fiabilidad o funcionalidad

Alvaro[64]: Es posible que algunas implementaciones no cubran completamente los servicios especificados. Este atributo mide el número de operaciones implementadas comparado al número total de operaciones especificadas.

Software Quality Assurance Group [65]: La característica del software que proporciona una implementación completa de las funciones requeridas.

Pressman[1]: El grado alcanzado en la implementación total de una función

Atributos relacionados: Corrección, portabilidad, confiabilidad, Funcionalidad, mantenibilidad, usabilidad

Modelos en los que aparece: McCall, Boehm, Dromey

CONSISTENCIA

Nombre en inglés: Consistency

También conocido como: Coherencia, Consistente (Dromey)

Descripción:

McCall [18]: Los atributos del software que proporcionan un diseño uniforme, técnicas de implementación y notación

Boehm [17]: Medida en la que existe uniformidad en notación, terminología, y dentro de la simbología propia. Externa: Medida en la que el contenido es trazable hacia los requerimientos

Otras definiciones:

Dromey [26]: Una forma estructural es consistente si su uso mantiene sus propiedades o funcionalidad y todos sus elementos contribuyen, y refuerzan sus objetivos conjuntos o efectos

SQAE [30]: ¿Están los productos del proyecto (código y documentación) construidos con un estilo uniforme para un estándar documentado?

Software Quality Assurance Group [65]: Las características del software que proporcionan técnicas y notación uniformes de diseño e implementación

Pressman[1]: El empleo de un diseño uniforme y de técnicas de documentación a lo largo del proyecto de desarrollo del software.

IEEE 610.12 [23]: El grado de uniformidad, estandarización y libertad de contradicción entre los documentos o partes de un sistema o componente.

Atributos relacionados: Corrección, confiabilidad, mantenibilidad, funcionalidad, usabilidad, portabilidad, reusabilidad

Modelos en los que aparece: McCall, Boehm, FURPS, Dromey, SQAE

ENCAPSULAMIENTO

Nombre en inglés: Encapsulation (Bansiya), Encapsulated (Dromey)

También conocido como: Encapsulado (Dromey)

Descripción:

IEEE 610.12 [23]: Una técnica de desarrollo de software, que consiste en aislar una función de sistema o un conjunto de datos y operaciones sobre esos datos dentro de un módulo, y proveer especificaciones precisas para el módulo.

Otras definiciones:

Dromey [26]: La manera como son utilizadas las variables puede tener un impacto significativo sobre la modularidad y por lo tanto la calidad independiente de los módulos, programas y sistemas

Bansiya [31]: Definido como el adjuntar de los datos y comportamiento dentro de una construcción simple. En diseño Orientado a Objetos la propiedad específicamente esta referida a clases de diseño que previenen acceso a declaraciones de atributos por su definición para ser privados, entonces protege la representación interna de los objetos

Atributos relacionados: Mantenibilidad, confiabilidad, portabilidad, reusabilidad, comprensibilidad, efectividad

Modelos en los que aparece: Dromey, Bansiya

FACILIDAD DE CAMBIO

Nombre en inglés: Changeability, Expandability (McCall), extendability (IEEE 610.12), Augmentability (Boehm), extendibility (Bansiya, FURPS)

También conocido como: Facilidad de cambio, Modificabilidad (Boehm), extensibilidad (Bansiya, FURPS).

Descripción:

ISO 9126 [15]: La capacidad del producto software para permitir una modificación especificada que debe ser implementada

McCall [18]: Atributos de software proporcionados para expansión de requerimientos de almacenamiento de datos o funciones computacionales.

Otras definiciones:

Quint2 [59]: Esfuerzo necesitado para modificación, remoción de fallos o para cambios del ambiente.

Boehm [17]: Medida en la que un producto software fácilmente es expandido adecuadamente en los requerimientos de almacenamiento de datos o componentes computacionales funcionales

Bansiya [31]: Referido a la presencia y uso de las propiedades en un diseño existente de tal manera que permite para la incorporación de nuevos requerimientos en el diseño

Software Quality Assurance Group [65]: El esfuerzo requerido para incrementar capacidad o rendimiento del software incrementando funciones actualizadas o adicionando nuevas funciones o datos. Las características del software que proporcionan la capacidad de expansión para funciones o datos.

Pressman[1]: El grado con que puede ampliarse el diseño arquitectónico, de datos o procedimental

IEEE 610.12 [23]: La facilidad con que un sistema o componente puede ser modificado para incrementarse almacenamiento o capacidad funcional.

Berander [35]: La capacidad de un producto software de permitir una modificación especifica a ser implementada.

Atributos relacionados: Mantenibilidad, Flexibilidad, abstracción, herencia, polimorfismo, soportabilidad

Modelos en los que aparece: ISO 9126, Quint2, McCall, Boehm, FURPS, Bansiya

CAPA 3: FC-EXCLUSIVOS-BAJO NIVEL

ACOPLAMIENTO

Nombre en inglés: Coupling

Descripción:

Bansiya [31]: Define la interdependencia de un objeto sobres otros objetos en un diseño. Esta es una medida del número de otros objetos que podrían tener ser accedidos por un objeto en orden para ese objeto o función correctamente

Otras definiciones:

IEEE 610.12 [23]: La forma y grado de independencia entre módulos software.

Atributos relacionados: Cohesión

Modelos en los que aparece: Bansiya

FACILIDAD DE APRENDIZAJE

Nombre en inglés: Learnability

Descripción:

ISO 9126 [15]: La capacidad del producto software para permitirle al usuario aprender su aplicación (como por ejemplo, control de operación, entradas, salidas).

Otras definiciones:

Quint2 [59]: Esfuerzo de los usuarios para aprender las aplicaciones del software (por ejemplo control, entrada y salida)

Atributos relacionados: Usabilidad

Modelos en los que aparece: ISO 9126, Quint2

OPERATIVIDAD

Nombre en inglés: Operability

También conocido como: Facilidad de operación

Descripción:

McCall [18]: Atributos de software que determinan operación y procedimientos concernientes con la operación del software

Otras definiciones:

ISO 9126 [15]: Esfuerzo de los usuarios para operación y control de operación

Quint2 [59]: La capacidad del producto software para permitirle al usuario su operación y control

Software Quality Assurance Group [65]: La característica del software que determina operaciones y procedimientos concernientes con operaciones del software y que proporciona entradas usables y salidas que pueden ser asimiladas.
Pressman [1]: Facilidad de operación de un programa

Atributos relacionados: Usabilidad

Modelos en los que aparece: McCall, ISO 9126, Quint2

Los atributos de calidad han sido clasificados en los espectros como fue mencionado en la sección 2.2, sin embargo el numeral actual muestra solamente aquellos que clasificaron en el espectro claro. Como complemento a los atributos existe la necesidad de medirlos, para ello la siguiente sección introduce algunas de las métricas encontradas a lo largo de este proyecto.

3.3. Las métricas de calidad

Para comenzar esta sección primero debe advertirse al lector una consideración importante, el proyecto inicialmente fue pensado para la evaluación del producto software orientado a objetos, sin embargo, al realizar la búsqueda de métricas, pudo verse que éstas no solo son aplicables a productos orientados a objetos sino que pueden llegar a trascender a productos desarrollados bajo otros paradigmas, en este caso puede ejemplificarse aquellas métricas que incluyen al usuario en la evaluación, para poder aplicar la métrica es muy importante tener en cuenta la disciplina en donde aplicar la métrica y el atributo que desea valorarse, pues por ejemplo la orientación a objetos es más visible en la disciplina de diseño en el paradigma orientado a objetos.

Cada atributo ha sido clasificado según lo comentado en el capítulo 2, igualmente aplica a métricas. Las métricas no siempre son encontradas junto al atributo, así como no siempre son claras en su aplicación. Éste punto muestra una parte de las métricas encontradas, que presentan mayor claridad frente a las otras. El resto de las métricas pertenecientes al marco pueden ser consultadas en el Anexo H.

Las métricas han sido nombradas con números (Ej. Métrica 1), para su fácil manejo pues existen diversos factores que pueden confundir al lector en el momento de usarlas, estos son por ejemplo métricas con nombres similares pero contenido y definición diferentes,

métricas con nombre distinto pero que en esencia son lo mismo, entres otros. En los casos en donde son métricas relacionadas es colocada la referencia a las métricas en la sección de *otros.*

CAPA 4: MÉTRICAS

MÉTRICA 17

Nombre en español: Número de hijos o descendientes (NDD)

Nombre en inglés: Number of children (NOC)

Descripción: La métrica es el número de subclases de una clase en la jerarquía. Es un indicador del nivel de reutilización, la posibilidad de haber creado abstracciones erróneas, y es un indicador del nivel de pruebas requerido. Un mayor número de hijos requiere más pruebas de los métodos de esa clase y aumenta la dificultad para modificar la clase pues afecta a todos los hijos de dicha clase. Clases en un nivel más alto en la jerarquía deberían tener más subclases que las clases en un nivel más bajo en la jerarquía. NOC puede ser también un indicador del uso inadecuado de la herencia. Es un potencial indicador de la influencia que una clase puede tener sobre el diseño del sistema. Si el diseño depende mucho de la reutilización a través de la herencia, quizás sea mejor dividir la funcionalidad en varias clases.

Fórmula:

NOC= Numero de subclases inmediatamente subordinadas de una clase en la jerarquía de clases,

Atributo al que ayuda a calcular: Reusabilidad, herencia

Influencia sobre otros atributos: Analizabilidad, cambiabilidad

Granularidad: Clase

Rango: 0 hasta N, donde N es un entero positivo

Referencias: Olmedilla [53], Vázquez [61], Chidamber y Kemerer [67], González [68], Pressman[1], Pritchett [69], Basili [70], El-Wakil [71], Li [72], Lindroos [73], Andersson [74], Bar [75], Kan [24], Laird [76], Fenton [77], Garzas [57], Marin [28]

Espectro: Clara

Otros: aplica en disciplina de diseño. Es la misma métrica de NOD Number Of Descendants (Métrica 135)

MÉTRICA 20

Nombre en español: Acoplamiento entre objetos

Nombre en inglés: Coupling Between objects (CBO)

Descripción: Es el número de clases a las cuales una clase está acoplada, sin tener con ella relaciones de herencia (hijo o padre). Hay dependencia entre dos clases cuando una de ellas usa métodos o variables de la otra clase. Es consistente con las tradicionales definiciones de acoplamiento: "medida del grado de interdependencia entre módulos". Es un indicador del esfuerzo necesario para mantenimiento y pruebas (un objeto esta acoplado a otro si una de sus acciones esta en el otro).

Fórmula:

CBO=Numero de acoplamientos de una clase

Atributo al que ayuda a calcular: Reusabilidad, acoplamiento

Influencia sobre otros atributos: complejidad, modularidad, encapsulamiento, Analizabilidad, cambiabilidad, estabilidad

Granularidad: Clase, programa, sistema

Referencias: Vázquez [61], Chidamber y Kemerer [67], T22-garzas, Olmedilla [53], A42-Manso, Basili [70], El-Wakil [71], , Lindroos [73], Andersson [74], Bar [75], Laird [76], Fenton [77], Marin [28], Ojha [78]

Espectro: Claro

Otros: Aplicado en la disciplina de diseño. Es sugerido como un indicador de esfuerzo para mantenimiento y pruebas. Cuanto más independiente es un objeto es más fácil de reutilizar. La métrica también puede verse como el número de mensajes enviados entre dos clases. Tiene relación con la métrica DAC (Métrica 28)

MÉTRICA 26

Nombre en español: Factor de acoplamiento

Nombre en inglés: Coupling Factor - COF

Descripción: Proporción entre el número real de acoplamientos no imputables a herencia y el máximo número posible de acoplamientos en el sistema. Es decir indica la comunicación entre clases.

Fórmula:

$$COF = \frac{\sum_{i=1}^{TC} \left(\sum_{j=1}^{TC} (es\_cliente(Ci, Cj)) \right)}{(TC^2)-TC}$$

Donde

(TC^2)-TC: es el máximo número de acoplamientos en un sistema con TC clases,

es_cliente (Ci, Cj): es 0 o 1, 1 (si y solo si Cc->Cs Cc<>Cs), 0 en caso contrario

La relación cliente-servidor (Cc -> Cs) significa que Cc (la clase cliente) contiene al menos una referencia no basada en la herencia a una característica (método o atributo) de la clase Cs (clase proveedora). El numerador representa el número real de acoplamientos no imputables a la herencia. El denominador representa el máximo número posible de acoplamientos en un sistema con TC clases. Las relaciones cliente servidor

Pueden tener distintas formas:

Paso de mensajes regular.

Paso de mensajes "forzado".

Iniciación y destrucción de objetos.

Asociaciones semánticas entre clases con una cierta relación (p.e. 1: 1, 1: n o n: m).

Atributo al que ayuda a calcular: Acoplamiento

Influencia sobre otros atributos: Medida indirecta de: complejidad, encapsulamiento, reutilización, extensibilidad, mantenimiento, analizabilidad, cambiabilidad, estabilidad

Granularidad: Clase, programa, sistema

Referencias: Vázquez [61], Abreu y Melo[79], Pressman [1], Champeaux [80], Marin [28]

Espectro: Claro

Otros: Aplicado en disciplina de diseño.

MÉTRICA 112

Nombre en español: Proporción de métodos ocultos

Nombre en inglés: Method hiding factor- MHF

Descripción: La métrica es la proporción entre la suma de los grados de invisibilidad de los métodos en todas las clases y el número total de métodos definidos en el sistema. MHF es la proporción entre los métodos definidos como protegidos o privados y el número total de métodos. MHF es propuesto como una medida de encapsulamiento, cantidad relativa de información oculta. Cuando MHF es incrementado, la densidad de defectos y el esfuerzo necesario para corregirlos debería disminuir. Los métodos heredados no son considarados.

Fórmula:

$$MHF = \frac{\{\sum \text{ para } i=1 \text{ hasta TC de } [\sum \text{ para } m=1 \text{ hasta Md } (Ci) \text{ de } (1-V(Mmi))]\}}{\{\sum \text{ para } i=1 \text{ hasta TC de } [Md(Ci)]\}},$$

Donde:

$$V(Mmi) = \frac{\{\sum \text{ para } j=1 \text{ hasta TC de } [es\_ \text{ visible } (Mmi, Cj)]\}}{TC},$$

es_ visible (Mmi, Cj) = (1 si y solo si Cj puede llamar a Mmi, 0 en caso contrario).

TC: es el número total de clases,

Md (Cj): número de métodos definidos en la clase Ci (no heredados),

V (Mmi): es la visibilidad: porcentaje total de clases desde las cuales el método Mmi es visible.

Atributo al que ayuda a calcular: Encapsulamiento

Granularidad: Clase

Referencias: Vázquez [61], Abreu y Melo[79], Pressman [1], Champeaux [80], Marin [28]

Espectro: Claro

Otros: Aplicado en disciplina de implementación

Al igual que los atributos, también las métricas han sido clasificadas en los espectros mencionados en la sección 2.2, con espectro claro.

3.4. Las heurísticas

Las heurísticas de calidad de software son aplicadas en diferentes etapas del desarrollo del producto, en particular las disciplinas relacionadas al proyecto. Por esta razón, a continuación es mostrada cada una de las heurísticas y su relación con una disciplina en particular. De manera similar a la relación con métricas, los atributos de calidad tienen

relación con las heurísticas en razón a los aspectos que resalta cada uno. Para una vista completa de las heurísticas consultadas, remítase al Anexo I.

Para el caso particular de las heurísticas han sido nombradas con números (Ej. Heurística 1), para su fácil manejo pues no siempre tienen un nombre o este no es suficientemente claro, el manejo numerado además permite llevar una cuenta de heurísticas.

A continuación está un conjunto de las heurísticas extraídas a partir de la aplicación de los criterios de selección de las mismas, mostrados en la sección 2.5.1, Criterios de clasificación de las heurísticas de calidad.

CAPA 5: HEURÍSTICAS

HEURÍSTICA 001

Nombre: Heurística basada en NOS (Número de pasos)

Descripción:

Los casos de uso demasiado cortos suelen ser incompletos o no representan realmente una interacción actor–sistema resultando generalmente triviales. Por otro lado, los casos de uso demasiado largos son frecuentemente incomprensibles, presentan un nivel de detalle excesivo [Lilly 1999] o avanzan a un ritmo muy lento [Cockburn 2001], es decir, están dirigidas con demasiada lentitud a alcanzar el objetivo del caso de uso siendo más difíciles de leer.

Rango recomendado: [4,9].

Disciplina: Análisis.

Atributos relacionados: Comprensibilidad, Documentación, Testeabilidad.

Referencia: Bernárdez [81]

HEURÍSTICA 011

Nombre: Abstracciones consistentes con la forma

Descripción:

La abstracción es la capacidad de interconectar con un concepto mientras seguramente ignora algunos de sus detalles- manejando diferentes detalles a diferentes niveles. Las clases base son abstracciones que permiten enfocarse sobre atributos comunes de un conjunto de clases derivadas e ignora los detalles

de clases especificas mientras está trabajando sobre la clase base. Una buena clase interfaz es una abstracción que permite enfocarse sobre la interfaz sin necesidad de preocuparse sobre el trabajo interno de la clase. La interfaz para una rutina bien diseñada provee el mismo beneficio a un bajo nivel de detalle, y la interfaz para un paquete bien diseñado o subsistema provee dicho beneficio a un alto nivel de detalle. Desde un punto de vista de la complejidad, el principal beneficio de la abstracción es que le permite ignorar detalles irrelevantes

Disciplina: Análisis, Diseño.

Atributos relacionados: Mantenibilidad, Reusabilidad, Testeabilidad.

Referencia: McConnell [66]

HEURÍSTICA 015

Nombre: Secretos y el derecho a la privacidad.

Descripción:

Una tarea clave en el diseño de una clase es decidir cuáles características deberían ser conocidas fuera de la clase y cuales de las restantes en secreto. Una clase debería usar 25 rutinas y exponer solamente 5 de ellas, usando las otras 20 internamente. Una clase debería usar varios tipos de datos y exponer ninguna información acerca de ellos. Este aspecto del diseño de clases es conocido como "visibilidad" desde lo que esta tiene que hacer con aquellas características de la clase que están "visibles" o "expuestas" al exterior de la clase.

Disciplina: Diseño.

Atributos relacionados: Mantenibilidad, Reusabilidad, Testeabilidad.

Referencia: McConnell [66]

HEURÍSTICA 019

Nombre: Anticipar diferentes grados de cambio.

Descripción:

Una buena técnica para identificar áreas probables a cambiar es primero identificar el subconjunto mínimo del programa que debería ser usado por el usuario. El subconjunto constituye el núcleo del sistema y es poco probable de cambio. Pueden ser bastante pequeños que parezcan triviales. Esas áreas de mejora potencial constituyen cambios potenciales en el sistema; diseña dichas áreas usando los principios de ocultamiento de la información.

Disciplina: Mantenimiento.

Atributos relacionados: Mantenibilidad, Reusabilidad.

Referencia: McConnell [66]

HEURÍSTICA 036

Nombre: Heurística 2.5

Descripción:

No coloque detalles de implementación tales como funciones privadas de código compartido en la interfaz pública de una clase.

Disciplina: Diseño, Implementación.

Atributos relacionados: Mantenibilidad, Reusabilidad, Testeabilidad.

Referencia: Riel [82]

Como puede verse, las heurísticas tienen relación con los atributos de acuerdo a la característica del producto con que puede estar asociada. Dicha relación es explicada en el siguiente numeral.

3.5. Relaciones existentes

Para los atributos, métricas y heurísticas, éstas presentan relaciones de dos tipos: atributo-métrica y atributo-heurística. Estas relaciones son ampliadas a continuación (ver Figura 16).

Relación Atributo-Métrica: relación presente principalmente a través de un sub-atributo, pues el atributo generalmente no está relacionado directamente con las métricas, es una relación evaluativa, puesto que las métricas son medidas del sub atributo, con las cuales espera obtenerse una valoración del nivel de cumplimiento del atributo a las expectativas del usuario. Normalmente una métrica es usada cuando ya tiene al menos una parte del producto para que pueda ser evaluado.

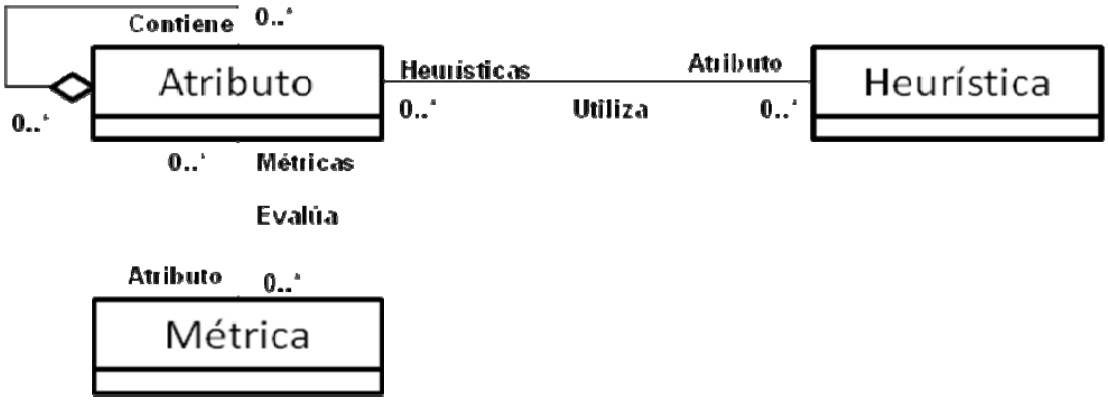

Figura 16. Relación Atributo-Métrica

Relación Atributo-Heurística: relación presente en la mayoría de casos de forma directa para los sub atributos de calidad, es una relación de tipo observación, puesto que las heurísticas tratan de sugerir buenas prácticas u observaciones a ser usadas para que alcance un mejor nivel de cada atributo, es decir que cumpla con lo esperado. Normalmente las heurísticas son usadas antes de comenzar a implementar un producto o entre disciplinas y/o iteraciones para mejorar los resultados.

Las relaciones de atributo con métrica o heurística, no necesariamente están conectadas entre sí, sin embargo tanto métrica como heurística tiene algún tipo de influencia sobre el atributo ya sea de valoración o de observación. Además existe la relación entre atributo y sub atributo la cual es mencionada en la siguiente sección.

3.6. Las influencias

Además de las relaciones de atributo-métrica y atributo-heurística, existe una relación que no debe ser olvidada, y es la existente entre Atributo y sub atributo. Esta puede verse como una relación de influencia de los atributos internos o sub atributos hacia los atributos de alto nivel o externos, en donde las métricas ayudan a valorar el sub atributo, que contribuye en parte a la evaluación del atributo. De esta manera muchos modelos de calidad han utilizado una estructura tipo jerarquía para representar esta relación, la cual será retomada en este trabajo (Anexo J), la siguiente tabla (Tabla 9) muestra un ejemplo de lo anterior

Usabilidad	Amigabilidad	
	Asistencia de Usuario	
	Atractivo	
	Auto-Descripción	
	Claridad	
	Completitud	
	Comunicatividad	
	Confiabilidad	
	Conformidad	
	Consistencia	
	Documentación	
	Eficiencia	
	Comprensibilidad	Auto-Descripción
		Cohesión
		Concisión
		Encapsulamiento
		Legibilidad
	Facilidad de aprendizaje	
	Ingeniería humana	
	Operatividad	
	Personalización	

Capa1 Capa 2 Capa3 Compartido Capa 3 Exclusivo

Tabla 9. Relación de Usabilidad con otros atributos y sub-atributos

La relación de influencia dada entre los atributos, está representada entre las capas del marco, dicha relación está basada en la estructura jerárquica propuesta en otros modelos como McCall y Boehm. En este punto son considerados los atributos encontrados y sus relaciones, tomando en cuenta los modelos en los que aparecen; sin embargo la Tabla 9 sólo consideró aquellos atributos clasificados en el espectro claro y gris.

Como muestra la Tabla 9, un atributo puede ayudar a atributos de su misma capa o superiores, es decir en caso de no existencia de la métrica para un atributo puede hacerse uso de uno de los atributos que ayuda a su cálculo para llegar a un resultado. El marco en este punto ha presentado los elementos básicos cumpliendo así con el objetivo de conformar el compendio, sin embargo existe la necesidad de mostrar algunos elementos adicionales para que el marco sea entendido mejor, dando lugar al capitulo 4.

CAPÍTULO IV: ELEMENTOS ADICIONALES AL MARCO CONCEPTUAL

En el presente capítulo describe algunos elementos considerados importantes para completar el marco conceptual de atributos métricas y heurísticas de calidad. Primero haciendo una consideración adicional concerniente con las relaciones entre los elementos del marco, segundo contempla algunas recomendaciones, y finalmente plantea una guía, representada en un diagrama de actividad, para poder hacer uso de la información presentada en este trabajo. Este capítulo contribuye a cumplir con los objetivos de plantear recomendaciones y un proceso preliminar de verificación de resultados del marco conceptual.

4.1. Relaciones planteadas

A partir de las relaciones mencionadas el capitulo anterior, la Figura 17 sugiere la relación entre atributos, métricas, heurísticas y recomendaciones (siguiente sección). Dicha relación permite ver la manera en que un elemento influye sobre otro, esto es, de un lado, la presencia de una relación directa entre el atributo y la métrica y, por otro lado, entre atributo y heurística.

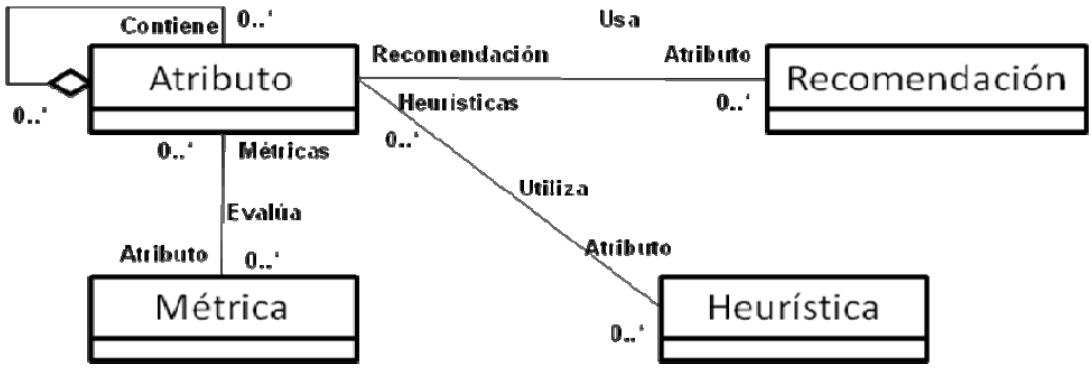

Figura 17. Relación Métrica-Atributo-Heurística y Recomendación

En este caso, son retomadas las relaciones que ya existen, presentadas en la sección 3.5, de manera que las recomendaciones planteadas tengan peso sobre los tres elementos del marco, dependiendo del momento en que sean aplicadas. Es decir, si las recomendaciones son tenidas en cuenta, por ejemplo desde el comienzo del desarrollo de

un proyecto o antes de comenzar con una disciplina, los resultados estarán reflejados en el uso de heurísticas, que podrán influir en la calidad de un atributo y en consecuencia la aplicación de la métrica arrojará resultados favorables.

4.2. Recomendaciones

Surgen a partir de dos aspectos: por un lado, la evaluación de una (o un conjunto) métricas y, por otro lado, la orientación que prestan las heurísticas, todas las anteriores son originadas desde la aplicación de un atributo de calidad, o un conjunto de ellos, a un producto software.

Para una mejor comprensión, las recomendaciones están agrupadas por disciplina. A continuación puede observarse un subconjunto de recomendaciones que surgen a partir de tres elementos principales: atributos, métricas y heurísticas.

RECOMENDACIÓN 01

Descripción:

Los casos de uso demasiado cortos suelen ser incompletos o no representan realmente una interacción actor–sistema resultando generalmente triviales. Por otro lado, los casos de uso demasiado largos son frecuentemente incomprensibles, presentan un nivel de detalle excesivo, siendo más difíciles de leer. Un rango recomendado de pasos al interior de caso de uso debería ser entre 4 y 9 pasos.

Disciplina: Análisis.

RECOMENDACIÓN 10

Descripción:

Una tarea clave en el diseño de una clase es decidir cuáles características deberían ser conocidas fuera de la clase y cuales de las restantes permanecerán ocultas. Una clase debería usar 25 rutinas y exponer solamente 5 de ellas, usando las otras 20 internamente. Es recomendado que, dada una clase, ésta debería usar varios tipos de datos y no exponer información alguna acerca de ellos. Este aspecto del diseño de clases es conocido como “visibilidad” desde lo que esta tiene que hacer con aquellas características de la clase que están “visibles” o

"expuestas" al exterior de la clase. Esto evita manejar clases grandes y poco cohesionadas.

Disciplina: Diseño.

RECOMENDACIÓN 32

Descripción:

Es recomendado que el personal de pruebas tenga la mayor información sobre el software a probar, de manera que sea capaz de entender bien el diseño, los cambios en el diseño y las dependencias entre componentes internos, externos y compartidos.

La documentación técnica debería ser accesible al instante, precisa, bien organizada, específica y detallada.

Disciplina: Pruebas.

RECOMENDACIÓN 33

Descripción:

Es recomendado que, para una jerarquía de herencia, las dependencias de la clase descendiente deberían ser una consideración principal en la elaboración de cualquier decisión de refactorización.

Disciplina: Mantenimiento.

RECOMENDACIÓN 34

Descripción:

Es recomendado que, para lograr un alto nivel de abstracción, haga uso de diagramas como una herramienta heurística poderosa que logra buenas representaciones de gran variedad de aspectos en diferentes instancias del transcurso del proyecto. Recuerde que una imagen vale más que mil palabras.

Disciplina: Análisis y Diseño.

RECOMENDACIÓN 37

Descripción:

En ocasiones, es recomendado que aquellas propiedades comunes podrían ser una abstracción única, una clase y una instancia. Esto hace la propiedad más centralizada y reutilizable.

Disciplina: Diseño e Implementación.

RECOMENDACIÓN 46

Descripción:

Es recomendado que las propiedades de la clase permanezcan ocultas detrás de métodos de acceso. Esto significa que las propiedades de la clase no deberían ser usadas en la interfaz pública de una clase. La interfaz pública de una clase es una lista de métodos y atributos que podrían ser utilizados desde el exterior de la clase.

Disciplina: Diseño y Mantenimiento.

RECOMENDACIÓN 48

Descripción:

Buscando consistencia en el sistema, es sugerido que éste debe prevenir que los usuarios tengan que preguntarse si las diversas palabras, situaciones, o acciones significan la misma cosa. La consistencia puede darse de las siguientes maneras:

a. Visual: Los elementos de la interfaz de usuario deben ser consecuentes o consistentes en el aspecto y estructura.

b. Funcional: La manera de llevar a cabo las diferentes tareas a través del sistema debe ser consecuente, con otros sistemas similares, e incluso entre diferente clases de aplicaciones en el mismo sistema.

c. Evolutiva: Debe ser fácil tener una nueva versión del software, y que esta sea consistente con las anteriores. Como por ejemplo, en el caso de una familia de producto software, la consistencia sobre los productos en la familia es un aspecto importante.

Disciplina: Pruebas y Mantenimiento.

Las recomendaciones son observaciones hacia los usuarios, para que de esta forma tenga a bien considerar buenas prácticas a sus desarrollos de software, si estas son aplicadas desde un principio podrán tener influencia sobre la arquitectura, y por ende mejorar la calidad. El resto de recomendaciones pueden consultarse en el Anexo K. El proyecto además ha pensado en una guía para el uso de éste y los otros elementos del marco, el cual es presentado en la siguiente sección.

4.3. Guía para utilizar el marco conceptual.

Para poder hacer uso del marco conceptual, es importante plantear una guía a seguir, que fue pensada partiendo de trabajos como FURPS [20], ISO 9126 [15] y otras guías [25][83][93] (Ver Figura 18).

El diagrama planteado tiene los siguientes pasos (ver Figura 18):

1. Inicialmente propone que el usuario del marco haga una selección de los atributos de calidad que le sean más convenientes a usar para su producto. Adicionalmente seleccionar aquellos sub-atributos relacionados que sean más cercanos a sus intereses

2. Seguidamente seleccionar las métricas relacionadas al sub atributo, con las cuales debe realizarse la medición. También seleccionar las heurísticas que podrían servir a los atributos seleccionados.

3. Dependiendo de la necesidad de aplicación escoger entre artefacto, iteración o incremento del producto software a desarrollar, y preguntar si será evaluado: si es así, continuar al punto 4, si no, seguir con el paso 3.1:
   3.1. Aplicar las heurísticas seleccionadas a la iteración o disciplina empleada.
   3.2. Volver al punto 3.

4. En caso de que sea un artefacto, iteración o incremento que entra a ser evaluado:
   4.1. Aplicar las métricas.
   4.2. Ver si el resultado obtenido con las métricas es el esperado. Si no es el resultado esperado continuar al punto 4.3, si el resultado esperado continuar al punto 4.4.
   4.3. Si no es el resultado esperado:
      4.3.1. Consultar las recomendaciones
      4.3.2. Aplicar los cambios que sean necesarios.
      4.3.3. Aplicar las métricas nuevamente hasta que los resultados sean los esperados o cercanos
   4.4. Si son los resultados esperados, terminar.

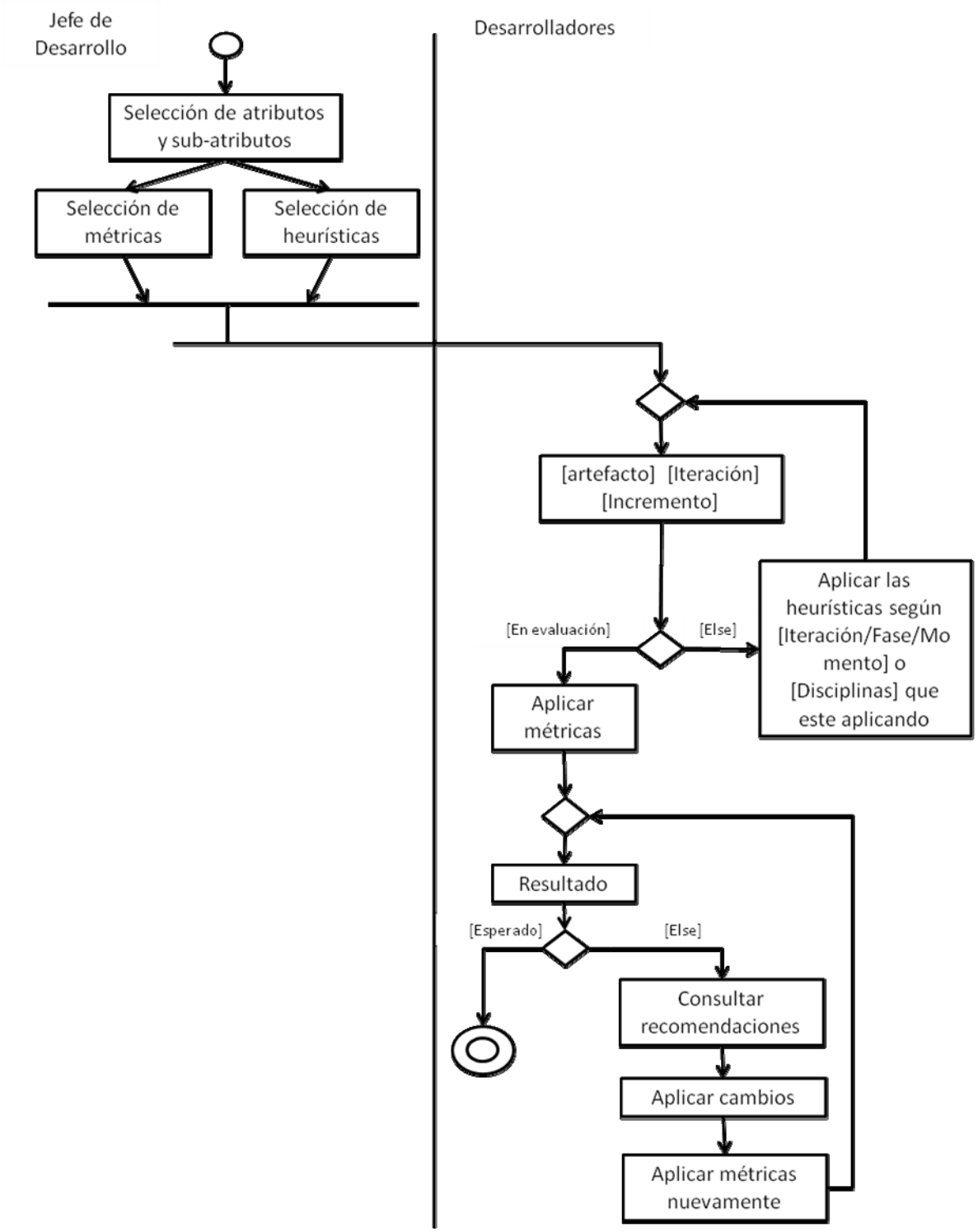

Figura 18. Diagrama de actividad del marco conceptual

El diagrama de actividad que representa la guía, fue pensado de forma que pueda usarse en casi cualquier etapa de desarrollo y/o artefacto software. Tomando en cuenta los elementos del marco conceptual y la guía de uso fue realizada una experiencia en empresas de la región sur-occidental colombiana, la cual es plasmada en el capítulo 5

CAPÍTULO V: EXPERIENCIA CON LAS EMPRESAS

A partir de lo expuesto en los capítulos anteriores, este capítulo expresa la experiencia realizada con MIPYMEs productoras de software del sur-occidente Colombiano, desde su caracterización hasta la aplicación de una muestra del marco conceptual, contribuyendo así al objetivo de plantear un proceso preliminar de verificación de resultados del marco conceptual, apoyándose en muestras de empresas de software de la región.

5.1. Empresas

En el desarrollo del trabajo con las empresas participó personal de cada una de ellas, a fin de obtener la información necesaria que facilitara las actividades a realizar en este proyecto.

Para obtener información de las empresas y sus características fue utilizado el recurso de las encuestas, aplicadas a cuatro empresas de la región localizadas específicamente en las ciudades de Popayán en el Cauca, y Pasto en Nariño, dos empresas en cada ciudad. Para contactarlas inicialmente los responsables fueron buscados en Internet, siendo finalmente empresas afiliadas a PARQUESOFT, previa confirmación de existencia en cámara de comercio. Debido al tamaño de las empresas las encuestas fueron realizadas por una o dos personas en cada empresa. A continuación puede verse la caracterización de cada empresa.

5.1.1. Caracterización

Las empresas con las cuales fue llevado a cabo el trabajo fueron: Latin-Business, SITI, Input y Seratic. A continuación puede observarse una breve caracterización de ellas:

- Empresa Nº 1:
  Nombre: Latin Business Ltda.
  Año de constitución: 2004
  Acerca de la empresa: Empresa constituida en el 2004 por un grupo de profesionales que unen sus conocimientos e integran sus capacidades de

innovación en torno al desarrollo de sistemas informáticos, de telecomunicaciones y de desarrollo de software, mediante el uso de herramientas de vanguardia. Dichos sistemas están ajustados a las necesidades y requerimientos de sus clientes, garantizando alta calidad, respaldo, seguridad y confiabilidad en el desarrollo de sus procesos.

La empresa está especializada en la implementación de Portales y sitios Web a través del uso del CMS Joomla, el cual cuenta con un alto grado de desarrollo, y un proceso de actualización permanente. Este CMS permite la instalación de múltiples componentes y módulos para tareas especificas, adaptables a las necesidades de cada cliente. Tiene experiencia en los mecanismos de seguridad que deben tenerse en cuenta cuando trabajan con Joomla y con sus diversos componentes, para garantizar un funcionamiento completo de sus sitios.

- Empresa Nº 2:
  Nombre: SITI Ltda.
  Año de constitución: 2002
  Acerca de la empresa: Empresa constituida en el 2002, especializada en desarrollo tecnológico que tengan las empresas e instituciones según sea el campo. Cuenta con experiencia en el área de sistemas, trabajando en desarrollo de software sobre diferentes plataformas reconocidos a nivel mundial como PHP, VTCL, Delphi, PostgreSQL, MySQL, ORACLE. Ofrece asesoría necesaria de acuerdo a las necesidades y brinda respaldo para que la información siempre este segura, gracias al manejo de plataformas tales como Linux, Microsoft Windows, Sun Solaris entre otras. Además, busca solucionar problemas de comunicaciones a través de la implementación de redes inalámbricas o de cableado estructurado.

  La empresa busca solucionar necesidades específicas de sistematización para empresas, creando herramientas que permitan mantener un óptimo manejo de la información a partir del desarrollo de sitios Web y portales empresariales, y además ofrece el servicio de desarrollo de software a medida. También ofrece capacitación en sistemas operativos, aplicaciones de oficina, manejo de herramientas especializadas y configuración de servidores Linux. Adicionalmente ofrece asesorías en sistematización, proyectos de desarrollo tecnológico,

administración de centros de cómputo, evaluación de proyectos, asesoría en adquisición de equipos, auditoria en sistemas y seguridad informática.

- Empresa Nº 3:
  Nombre: Input Technologies Ltda.
  Año de constitución: 2004.
  Acerca de la empresa: Empresa creada en el 2004, con el fin de brindar soluciones tecnológicas integrales que contribuyan al mejoramiento continuo de los procesos y faciliten la obtención de la calidad de las empresas en general, contando para ello con talento humano idóneo y comprometido con amplia experiencia en procesos de calidad.

- Empresa Nº 4:
  Nombre: Seratic Ltda.
  Año de constitución: 2005
  Acerca de la empresa: Empresa especializada en proveer soluciones en Tecnologías de Información y Comunicación (TICs), para resolver necesidades de movilidad en los procesos clave de las organizaciones. Provee soluciones disponibles para una variedad de dispositivos móviles, que permitan tratar la información de forma segura y confiable, enfocadas a incrementar la productividad reduciendo tiempos de operación, la eficiencia en la ejecución de los procesos, y también incrementar cobertura de atención u operación, de manera que pueda controlar y supervisar actividades remotas.

  De igual manera, las soluciones móviles que proveen, soportan y complementan el uso de plataformas Web para la generación de reportes, monitoreo y seguimiento de procedimientos y resultados, análisis de tiempos, gestión de usuarios, permisos, roles, configuraciones y demás.

Es importante resaltar el interés y la preocupación de las empresas por mejorar día tras día en el desarrollo de sus productos y la manera de hacerlo, a través de diversos métodos, metodologías y con el uso de herramientas para ello. Sin embargo por tratarse de organizaciones pequeñas, con poco presupuesto, dificulta el acceso que éstas puedan tener a información para hacer mejoras en la calidad de sus productos software.

Es posible ver que las MIPYMEs productoras de software existentes en el sur-occidente Colombiano son empresas jóvenes con experiencia y personal limitado, sin embargo han hecho un gran esfuerzo para sobresalir en la región y fuera de ella, trabajando en áreas diversas. En este sentido, debido a sus características operativas, dejan muy poco tiempo o ninguno, para la aplicación de técnicas que mejoren la calidad en sus productos software, como es posible observar a partir de la información obtenida desde el Anexo L hasta el Anexo O.

Después de hacer la caracterización de la empresa, la información adquirida de las mismas encuestas es usada para conformar un compendio de atributos y métricas con la cual valorar la calidad de un modulo o artefacto de un producto suministrado por ellas mismas.

5.2. Compendio

Comprende la información obtenida a partir del análisis del material recopilado. Dicha información representa aquella que inicialmente fue consultada y posteriormente depurada, con el objeto de recoger una parte representativa de los puntos de vista de las diferentes fuentes consultadas; y a la que seguidamente, aplicó los criterios de selección mencionados en el capítulo 2, de modo que realiza un filtro a la información inicial para obtener, hasta este punto, la que es acorde con las necesidades de información expresadas por quienes participan de éste proyecto.

El compendio pretende ser base de información concisa respecto a atributos, métricas y heurísticas de calidad mencionadas en el marco conceptual, tal que permita su aplicación en un entorno dado, esto es, un producto software en una empresa. De aquí que la mayoría de atributos seleccionados por las empresas son aquellos de frecuente utilización según las personas entrevistadas.

5.2.1. Origen de evaluación

A continuación son mostrados los esquemas usados, que permiten la selección de atributos, sub-atributos y métricas de calidad de software. Para mayor información de dichos esquemas, consulte el Anexo P.

Los atributos mostrados en la sección 3.2 y el Anexo P6, resultados generales de las empresas, fueron promediados después de realizar una encuesta, que trató de averiguar el conocimiento e interés de las empresas hacia los atributos de calidad que desearían aplicar y su relevancia, como puede verse en las Tabla 10 y Tabla 11.

Nombre Atributo	Promedio
Fiabilidad (Confiabilidad)	8,7
Funcionalidad	9,0
Usabilidad (Facilidad de uso)	9,5
Mantenibilidad (Facilidad de mantenimiento)	6,8

Tabla 10. Atributos del compendio

Nombre Sub-atributo	Promedio
Acoplamiento	6,0
Auto-Descripción	2,0
Encapsulamiento	6,5
Facilidad de aprendizaje	8,0
Polimorfismo	7,0
Seguridad (Seguridad interna)	4,0
Tamaño	5,3
Reusabilidad	7.6

Tabla 11. Sub-atributos del compendio

Un procedimiento similar fue realizado para las métricas, como puede notarse a continuación

5.2.1.1. Tabla de métricas

A continuación es posible observar un conjunto de métricas, clasificadas como claras y grises a partir de los criterios mencionados en la sección 2.3 del presente documento y las métricas del capítulo 3, además de estar relacionadas con los atributos de la sección

anterior. Para obtener información adicional respecto al resultado de la clasificación final de las métricas encontradas, remítase al Anexo H.

Numero de la métrica	Nombre en ingles	Nombre en español.
111	Data access metric (DAM)	Métrica de acceso de datos (DAM)
112	Method hiding factor- MHF	Proporción de métodos ocultos
113	Attribute hiding factor- AHF	Proporción de atributos ocultos
20	Coupling Between objects (CBO)	Acoplamiento entre objetos
26	Coupling Factor - COF	Factor de acoplamiento
17	Number of children NOC	Número de hijos o descendientes (NDD)
182	Access controllability	Capacidad de control de acceso
187	Data encryption	Encriptación de datos
285	Size2	
274	Number of classes -NOC	Numero de clases
180	Polymorphism Factor -POF	Proporción de polimorfismo -FP
300 a 310	Self- descriptness	Medidas de Boehm para auto descripción
311	Quantity of comments (Por modulo)	Cantidad de comentarios
230	Effectiveness of the user documentation and/or help system	Efectividad de la documentación de usuario y sistema de ayuda
231	Effectiveness of the user documentation and/or help system in use	Efectividad de la documentación de usuario y/o sistema de ayuda en uso

Tabla 12. Métricas del compendio

El compendio es aplicado sobre información de artefactos suministrados por las empresas, los resultados son mostrados en la siguiente sección.

5.3. Resultados

Esta sección muestra la selección final de atributos, sub-atributos y métrica, que permiten ser aplicados a un artefacto en las empresas. Los artefactos que las empresas han proporcionado para ser valorados con el compendio previamente seleccionado son:

- Organización: INPUT Technologies Ltda.– Popayán
  Representante: Ricardo Andrés Ledezma
  Producto: Compromiso
  Modulo: Actas

- Organización: Latin Business Ltda. – Pasto.

Representante: Jhon Jairo Ortiz

Producto: Truck Dispatch

Modulo: Controladores

- Organización: SERATIC Ltda.– Popayán

  Representante: Diego Chamorro

  Producto: Easy Sales

  Modulo: Módulo de Pedidos

- Organización: SITI Ltda. - Pasto

  Representante: Gelber Moran

  Producto: SIGE – Sistema de Información para la Gestión empresarial Modulo: Centro de Documentación

  Componente: Correspondencia de documentos de entrada

Después de aplicar las métricas a los artefactos otorgados por cada empresa, es presentado el siguiente cuadro (Tabla 13) en donde puede verse la información obtenida tras el análisis de los resultados mostrados en el anexo Q.

Plantilla: Atributo, sub atributos y métricas									
Atributo	**Sub-atributo**	**Métrica**	**V. Mín**	**V. Máx**	**SERATIC**	**INPUT**	**LATIN**	**SITI**	**Promedio**
Confiabilidad	Encapsulamiento	Métrica de acceso de datos (DAM)	0	1	0,7	0,3	0,3	1	0,53
		Method hiding factor- MHF	0	1	0,3	NA	0,1	0,06	0,08
		Attribute hiding factor- AHF	0	1	0,2	0,6	0,04	0,4	0,35
	Acoplamiento	Coupling Between objects (CBO)			18,5	2,5	4,1	2,1	2,90
		Coupling Factor - COF	0	1	0,05	0,1	0,3	0,09	0,16
Mantenibilidad	Encapsulamiento	Métrica de acceso de datos (DAM)	0	1	0,7	0,3	0,3	1	0,53
		Method hiding factor- MHF	0	1	0,3	NA	0,1	0,06	0,08
		Attribute hiding factor- AHF	0	1	0,2	0,6	0,04	0,4	0,35
	Reusabilidad	Coupling Between objects (CBO)			18,5	2,5	4,1	2,1	2,90
		Number of children NOC			3,2	0	3	1,2	1,40
Funcionalidad	Seguridad	Access controllability	0	1	1	1	0,7	1	0,90
		Data encryption	0	1	1	NA	1	1	1,00
	Tamaño	Size2			46,9	4,4	7,5	25,2	12,37
		Numero de clases			11	8	7	12	9,00
	Polimorfismo	Polymorphism Factor -POF	0	1	NA	NA	0,02	0	0,01
Usabilidad	Auto descripción	Medidas de boehm para auto descripción			0,3	0,2	0,5	0,625	0,44

		Quantity of comments (Por modulo)	0	1	NA	NA	NA	0,11	0,11
	Facilidad de aprendizaje	Effectiveness of the user documentation and/or help system	0	1	NA	NA	NA	0	0,00
		Effectiveness of the user documentation and/or help system in use	0	1	NA	NA	0,7	0,85	0,78

Tabla 13. Resultados de las empresas

Teniendo en cuenta los resultados de la Tabla 13, es posible extraer las siguientes conclusiones:

- El número total de métricas presentadas fue 15 y el número total de métricas depende del artefacto que la empresa suministra para realizar la evaluación.

- Para la empresa Latin Business Ltda., el número total de métricas aplicadas fue 13. Por tanto, el porcentaje de aplicación es de aproximadamente 87 por ciento, lo cual es considerable teniendo en cuenta la cercanía al máximo.

- Para la empresa SITI Ltda., el número total de métricas aplicadas fue 15, esto quiere decir que en el producto suministrado fue posible identificar la totalidad de aspectos que proveen las métricas encontradas, y además es un posible indicio de buenas prácticas encontradas en dicho producto.

- Para la empresa Input Technologies Ltda., el número total de métricas aplicadas fue 9. Por tanto, el porcentaje de aplicación es de aproximadamente 60 por ciento, indicando que algunas de las prácticas que sugieren las métricas no son realizadas en el producto suministrado.

- Para la empresa Seratic Ltda., el número total de métricas aplicadas fue 11. Por tanto, el porcentaje de aplicación es de aproximadamente 73 por ciento, lo cual es considerable teniendo en cuenta la cercanía al máximo.

En general, sí fue posible la aplicación del compendio en las empresas, esto debido a las características que presenta, tales como sencillez y objetividad. Cabe destacar que no requiere mayor esfuerzo la aplicación de los ítems a disposición; pero implica que el

personal, de parte de la empresa, que realiza el ejercicio, sea alguien que tenga suficiente conocimiento del producto a evaluar. Esto debido al enfoque con el que trabaja la parte del evaluador, dado que en muchos casos las empresas no están preocupadas a profundidad de los aspectos mencionados en el compendio.

CAPÍTULO VI. CONCLUSIONES, LECCIONES APRENDIDAS Y TRABAJO FUTURO

Este capítulo presenta las conclusiones obtenidas en la experiencia del marco conceptual de atributos, métrica y heurísticas de calidad del producto software orientado a objetos, desde la búsqueda y recolección de la información, creación de los criterios de selección para cada elemento, y la clasificación en capas, además de la experiencia realizada con las MIPYMEs productoras de software de la región También muestra las lecciones aprendidas durante el desarrollo del presente trabajo de grado y los trabajos que podrían realizarse en esta misma línea de investigación.

6.1. Conclusiones

A continuación son puntualizadas las conclusiones derivadas del presente trabajo de grado

- En la elaboración de este trabajo los siguientes inconvenientes fueron encontrados por los autores: (1) las ambigüedades entre las definiciones y nombres de los atributos presentados en cada modelo, (2) los atributos que presentan definiciones confusas e incompletas, (3) el conflicto causado por errores de traducción, por último (4) el desafío de darle consistencia al marco propuesto, mediante la concepción de criterios para la ubicación adecuada de algunos atributos de calidad.
- Debido al tamaño de las MIPYMEs productoras de software de la región, no es posible tener un grupo que asegure la calidad de sus productos, esto hace que una persona asuma varios roles dentro de la empresa y sus proyectos. Por esta razón el tiempo dedicado a la calidad es mínimo o no existe.
- Los modelos de calidad para producto software, en su mayoría, han sido inspirados en modelos anteriores, lo cual hace que tengan una gran posibilidad de encontrar divergencias y similitudes entre sus atributos de calidad, además de contemplar mejoras que los modelos anteriores no han tomado en cuenta. Es decir, los modelos más actuales para producto software, están basados en algunos ya existentes buscando especializaciones que permitan mejorar inconsistencias o refinar diversidad de aspectos mostrados en ellos.

- Los estudios de aplicación de los modelos son muy limitados al público en general, ya sea porque no existen publicaciones, o porque el acceso a dicha información está restringido a personal especifico.
- La información encontrada en las diferentes fuentes que hacen referencia a modelos, atributos, métricas y heurísticas no siempre es completa, debido a que, en principio, existe dificultad en acceder a las fuentes originales y, en segundo lugar, a continuos errores de otros autores que hacen referencia los modelos, como lo son cambios en los modelos base, en algunos casos cambiaban los nombres originales de los atributos, colocaban mas atributos de los que el modelo posee u omitían elementos. Esto dificulta el trabajo en el momento de comprender estos conceptos y sus relaciones.
- Cuando el marco ya estaba conformado por sus elementos iníciales como atributos métricas y heurísticas, el paso siguiente fue recurrir a las MIPYMEs (INPUT, Latin Business, SERATIC, SITI) para consultar cuáles de los atributos eran los que más conocían y les interesaba trabajar, de esta forma saber cómo podría conformarse un compendio para ser aplicado a ciertos sub productos suministrados por las mismas empresas. Una vez aplicado el compendio y obtenidos los resultados, pudo lograrse un porcentaje de aplicación promedio del 80%, a pesar esto, los resultados para las MIPYMEs no fueron tan positivos, lo que evidencia la baja aplicación de prácticas de calidad en sus productos software, sin embargo las empresas manifiestan su interés en el trabajo realizado, y la continuidad en su disposición de colaboración para posibles trabajos futuros. Las actividades planteadas para este proceso arrojaron los resultados plasmados en los anexos. Para ver el seguimiento de actividades, por favor refiérase al Anexo V.
- Ya existe un primer acercamiento entre los conceptos de atributo y métricas o heurísticas, a pesar que en la documentación revisada aparecían relaciones entre atributos y métricas o entre atributos y heurísticas, era difícil ver la colección y conexión de los tres elementos en un solo documento, en este sentido éste trabajo aporta un punto de vista más amplio en donde está contemplada la relación de los tres conceptos.
- En el transcurso del proyecto pudo encontrarse que algunos elementos del marco trascendían mas allá de lo esperado, en el caso especifico de las métricas, las cuales inicialmente fue pensado en integrar al marco solo aquellas para producto desarrollado bajo el paradigma orientado a objetos, mostrando que ellas pueden

aplicarse a productos desarrollados bajo otros paradigmas, por ejemplo en el caso de las métricas asociadas al atributo de usabilidad, que están más relacionadas a la interfaz y a las necesidades del usuario.

6.2. Lecciones Aprendidas

Ésta sección describe las lecciones aprendidas a lo largo del desarrollo del presente trabajo.

- Factores tales como costos, disponibilidad del material, idioma y localización, son determinantes para encontrar la información de modelos, atributos, métricas y heurísticas de calidad del producto software. En la mayoría de casos la información tiene un costo elevado lo cual dificulta su obtención, en otros casos está presente la barrera del idioma, pues mucha información está en ingles técnico u otros idiomas, finalmente algunos documentos no pueden ser adquiridos por Internet o comprados por otro medio, están en bibliotecas o universidades distantes (e. g. Estados Unidos).
- El uso de modelos de calidad para producto software ya establecidos y documentación relacionada ha sido de gran importancia para este trabajo, sin embargo es necesario tener cuidado al estudiarlos debido a que diversos documentos hablan de calidad y los modelos, pero en algunas oportunidades puede presentarse confusión pues cada autor toma solo los elementos que necesita, en este sentido es muy importante referirse en lo posible a fuentes primarias, para confirmar que la información encontrada es fiable.
- En el momento de hacer las encuestas debe tenerse claro cuál es la información a obtener, para evitar confusiones al encuestado y sus respuestas sean lo más precisas posible. Para ello es recomendado el uso de la encuesta estadística con formulación de preguntas cerradas o semi-cerradas y de evaluación.
- El primer acercamiento a las empresas fue difícil, debido principalmente a la disponibilidad para dedicar tiempo a la calidad y, en particular, en lo concerniente al presente trabajo de grado. Dicha situación ocasionalmente fue repetida en algunas empresas con el transcurso del desarrollo de éste proyecto, pero no fue

impedimento para la realización de todas las actividades del mismo que involucraran interacción con las empresas.

- El trabajo en las empresas es una oportunidad de conocer el estado actual de las MIPYMEs productoras de software en la región sur occidental colombiana, dando así un panorama general de cuál es el manejo de calidad que le dan a los productos software que desarrollan. En la situación específica de las empresas visitadas (INPUT, Latin Business, SERATIC, SITI), es posible observar que el tiempo y recursos dedicados a este tema son muy limitados o nulos, sin embargo, manifiestan tener un gran interés en mejorar dicha situación.
- La colaboración brindada por las empresas de Popayán y Pasto, fue de gran importancia, facilitando la retroalimentación de información por parte de las empresas, lo cual sirvió para centrar algunas de las inquietudes y necesidades que estas tenían en relación a la evaluación de calidad de sus productos software, y ayudó con el proyecto a tener un conocimiento de sus intereses con lo cual fue configurado el compendio aplicado; por esta razón fue importante el contacto constante con las empresas.

6.3. Trabajo Futuro

Finalmente, las posibilidades de trabajo futuro que ofrece el presente trabajo, son presentadas.

- Construir una ontología que reúna los conceptos mostrados en el presente trabajo y permita hacer una representación del conocimiento disponible a cualquier persona interesada en la información.
- Brindar soporte a la información presentada en el marco conceptual, a través de una herramienta informática que permita la automatización de los datos planteados en él, y también permita el fácil acceso por parte de las MIPYMEs, integrando tecnologías como agentes inteligentes que den soporte a las recomendaciones y ayuden con las heurísticas, y minería de datos además de sistemas expertos que ayuden al cálculo de las métricas y la organización de los atributos, considerando la posibilidad de que esté en la Web, y permita mayor accesibilidad y disponibilidad de la información recopilada en el marco.

- Ampliar el marco conceptual con elementos como patrones y anti patrones, de forma que exista un mayor soporte para las recomendaciones, y exista una mayor correspondencia con las heurísticas.
- Realizar el proceso de verificación formal del marco conceptual ampliando la muestra de empresas y los conceptos reunidos en él, continuando el trabajo con las empresas actuales, quienes manifiestan el deseo de seguir colaborando con este tipo de proyectos, tal que permita valorar la aplicabilidad del marco un entorno empresarial dado.
- Ampliar el marco conceptual, contemplando posibles aspectos por mejorar que deja plasmado este trabajo, como lo son aquellos atributos y métricas clasificados como oscuros, en este sentido es recomendable hacer una búsqueda más exhaustiva o plantear la solución para cada elemento.

BIBLIOGRAFÍA

[1] PRESSMAN, Roger. Ingeniería Del Software, Un Enfoque Práctico. McGraw - Hill. 5ª Edición, 2002.
[2] ICONTEC Instituto Colombiano de Normas Técnicas y Certificación. Sistemas de Gestión de calidad. Fundamentos y Vocabulario. NTC – ISO 9000. División de publicaciones. 2000. 37 p. Bogotá.
[3] JETTER, Andreas. Assessing Software Quality Attributes, with source code metrics. University of Zurich. 2006.
[4] SCALONE, Fernanda. Estudio Comparativo De Los Modelos Y Estándares De Calidad Del Software. Universidad Tecnológica Nacional, Facultad Regional Buenos Aires. 2006.
[5] SALANOVA, Pilar. Modelos de Calidad Web. Clasificación de Métricas. Escuela Técnica Superior De Ingeniería Informática.2006.
[6] MORENO, Jair. ANDRADE, Hugo. BOLAÑOS, Liliam. "Compilación de un Modelo para Evaluar Atributos de Calidad en Productos Software". Revista Enlace Informático, Universidad Del Cauca. 2007.
[7] FERNANDEZ, Verónica, HORMIGA, María, TULANDE, Aleyda. Marco De Referencia Centrado En La Arquitectura Para La Mejora De Características De Usabilidad En El Desarrollo De Aplicaciones Web Construidas Por MIPYMEs. Universidad Del Cauca. 2008
[8] MONTERO, Francisco. Integración De Calidad Y Experiencia En El Desarrollo De Interfaces De Usuario Dirigidos Por Modelos. Universidad De Castilla La Mancha. 2005.
[9] MORENO, Jair. BOLAÑOS, Liliam. NAVIA, Manuel. Informe: Encuesta para el diagnóstico respecto a prácticas del producto software. Septiembre, 2008.
[10] KRISHNAIAH, Paruchuri. VAN DIJK, Nicolette. Handbook of Statistics 7: Quality Control and Reliability. Elsevier Science Publishers. North Holland, Ámsterdam. 1988.
[11] SOMMERVILLE, Ian. Ingeniería Del Software. Addison Wesley. 7ª Edición. 2005
[12] GONZALES, Moises. Método Del Desarrollo Arquitectónico en Grupo. Instituto Politécnico Internacional, México. 2006.
[13] OLSINA, Luis. BERTOA, Manuel, LAFUENTE, Guillermo. Un Marco Conceptual para -la Definición y Explotación de Métricas de Calidad. 2002
[14] BASS, Len, et. al. Software Architecture in practice. Addison Wesley. 2ª Edición. 2003.
[15] ISO/IEC 9126- 2001, Information Technology. Part 1, Part 2, Part 3 and Part 4. The International Standard Organization. 2002
[16] IEEE 1061-1998, IEEE Standard for a Software Quality, Metrics Methodology .Software Engineering Standards Committee of the IEEE Computer Society.
[17] BOEHM, Barry, BROWN, John. LIPOW, Myron. MACLEOD, Gordon. MERRIT, Michael. Characteristics of Software Quality, North-Holland, N.Y., 1978
[18] McCALL, Jim A. RICHARDS, Paul K. WALTERS, Gene F. Factors in software quality. Nat'l Tech. Information Service, Vol. 1, Vol. 2 y Vol. 3, 1977.
[19] CALDERON, Shlomi. Modelos De Calidad Enfocados A La Usabilidad, Aplicados En Los Procesos De Desarrollo De Sistemas De Información. Universidad De Castilla-La Mancha, Escuela Politécnica Superior. Obtenida: 7 Abril 2008.

Disponible en: http://alarcos.inf-cr.uclm.es/doc/cmsi/trabajos/Shlomi%20Calderon%20-%20Modelos%20de%20Calidad%20Enfocados%20a%20la%20Usabilidad%20-%20Doc.pdf

[20] GRADY, Robert. CASWELL, Deborah. Software Metrics: Establishing a Company-Wide Program. Prentice-Hall, 1987.

[21] CABALLERO, Edgar. Mejora de la calidad del software en el entorno de microempresas de TI. Universidad Politécnica de Madrid. 2007

[22] FITZPATRICK, Ronan. "Software Quality: Definitions and Strategic Issues. Advanced Research Module". Staffordshire University, School of Computing Report. April 1996.

[23] IEEE 610.12 -1990, IEEE Standard Glossary of Software Engineering Terminology. The Institute of Electrical and Electronics Engineers.

[24] KAN, Stephen. Metrics and Models in Software Quality Engineering. Addison Wesley. 2ª Edición. 2002.

[25] Portal de la NASA. Documentación sobre SATC. Abril de 2008. [En línea] Disponible en: http://satc.gsfc.nasa.gov

[26] DROMEY, Geoff. "A Model for Software Product Quality", Software Quality Institute Griffith University. Australia. 1994

[27] ISO/IEC 14598-1. Information Technology - Software Product Evaluation – Parte 1: General overview. The International Standard Organization. 1995

[28] MARIN, Beatriz, CONDORI-FERNÁNDEZ, Nelly, PASTOR, Oscar. Calidad en modelos conceptuales: Un análisis multidimensional de modelos cualitativos basados en la ISO 9126. RPM-AEMES, Vol. 4. 2007

[29] CÔTÉ, Marc-Alexis, SURYN, Witold, MARTIN, Robert, LAPORTE, Claude. Envolving a corporate software quality assessment exercise: A migration path to ISO/IEC 9126. SQP, Vol. 6, Nº 3. 2004

[30] MARTIN, Robert, SHAFER, Lawrence. Providing a framework for effective software quality assessment: A first step in automating assessments. The First Annual Software Engineering & Economics Conference. MITRE Hayes Auditorium, Virginia. 1996

[31] BANSIYA, Jagdish, DAVIS, Carl. " A Hierarchical Model for Quality Assessment of Object-Oriented Designs, doctoral dissertation", The University of Alabama in Huntsville, 1997.

[32] BASILI, Victor, CALDIERA, Gianluigi, ROMBACH, Dieter. The goal question metric approach. Encyclopedia of Software Engineering, pp. 528-532, John Wiley & Sons, Inc.1994.

[33] PIEDRAHITA, Sebastián. Construcción de una herramienta para evaluar la calidad de un producto software. Universidad EAFIT. Departamento de ingeniería de sistemas. 2007

[34] ABRIL, Enrique. Procesos de las aplicaciones Web: Informe sobre la "calidad de las aplicaciones Web". 2007.

[35] MORAGA, Mª Ángeles, CALERO, Coral, MARTINEZ, Julián, POZUELO, Antonio, ESPADAS, María. "Estudio del portal de Castilla-lamancha.es". Conferencia IADIS Ibero-Americana. 2005

[36] BERANDER, Patrick, et. al. "Software quality attributes and trade-offs". Blekinge Institute of Technology. 2005. http://www.bth.se/besq

[37] DUMKE, Reiner, CUADRADO-GALLEGO, Juan, ABRAN, Alain. The SMPI model: A stepwise process model to facilitate software measurement process

improvement along the measurement paradigms. Otto-von-Guericke-Universität Magdeburg. 2007.

[38] INTECO. Guía de mejores prácticas de calidad de producto. Instituto Nacional de Tecnologías de la Información. 2008

[39] ISO/IEC 12207(E) / IEEE std 12207. International Standard – Systems and software engineering – Software life cycle processes. The International Standard Organization. 2008

[40] CALERO, Coral, RUIZ, Julián, PIATTINI, Mario. "Classifying web metrics using the web quality model". Emerald, Online information review. Vol. 29, Nº 3. Pág: 227-248. 2005

[41] DAVILA, Leticia, MEJIA, Pedro. "Evaluación de la calidad de software en sistemas de información en Internet". 2003.

[42] ISO/IEC 25000 International Standard – Software engineering – Software product Quality Requirements and Evaluation (SQuaRE) – Guide to SQuaRE. The International Standard Organization. 2005

[43] ISO/IEC 20000 International Standard – Information Technology – Service management- part 1: Specification. The International Standard Organization. 2005

[44] SORIA, Victor. Relaciones humanas: Curso de comportamiento en la organización. Limusa. 2002.

[45] CORNWELL, Peter. Reusable Component Engineering for Hard Real-Time Systems. University of York, UK. 1998.

[46] SIMON, Hebert. "Modelling human mental processes" Proc. (AFIPS). Western Joint Computer Conference, Vol. 19, 1961

[47] HENAO CALAD, Mónica. Herramientas para la gerencia del conocimiento. Universidad EAFIT. Medellín. Obtenida: 21 Abril 2008. Disponible en: http://atlas.eafit.edu.co:8001/servlet/SBReadResourceServlet?rid=1175636219140_102401 0182_2161.

[48] BASHAR, Muhammad. An Overview of Object Oriented Design Heuristics. Department of Computer Science, Umeå University, Sweden. 2005

[49] Portal de Micro, Pequeña y Mediana Empresa - MIPYMEs - Mincomercio Colombia – Definición, "Pyme, definición," Agosto 2008. [En línea]. Disponible: http://www.mipymes.gov.co/pyme/home.asp.

[50] ALEMAN, Fernando, Importancia de las MIPYMEs en las Aglomeraciones Empresariales. * Una estrategia para el desarrollo regional en Colombia. REVISTA

[51] CAPOTE, Johana, LLANTEN, Julián. Marco Conceptual para la implantación de Gestión del Conocimiento en un Programa de Mejora de Procesos Software en MIPYMEs DS. Grupo de Investigación IDIS - Investigación y Desarrollo. 2008.

[52] GILB, Tom, Principles of Software Engineering Management; Addison Wesley, 1987

[53] OLMEDILLA, Juan. Revisión Sistemática de Métricas de Diseño Orientado a Objetos. Universidad Politécnica de Madrid, Facultad de Informática. 2005. Obtenida: 9 Abril 2008. Disponible en: http://is.ls.fi.upm.es/doctorado/Trabajos20042005/Olmedilla.pdf

[54] DROMEY, Geoff. "Software product quality: theory, model, and practice". Griffin University. Australia, Brisbane, Technical Report. 1998

[55] PANOVSKI, Gregor. Product Software Quality. Techische Universiteit Eindhoven. 2008

[56] ZEISS, Benjamin, VEGA, Diana, SCHIEFERDECKER, Ina, NEUKIRCHEN, Helmut, GRABOWSKI, Jens. "Applying the ISO 9126 Quality Model to Test Specifications Exemplified for TTCN-3 Test Specifications".
[57] GARZAS, Javier, PIATTINI, Mario. Object-oriented designs knowledge: principles, heuristics and best practices. IGI Global. Edición. 2006
[58] B. Van Zeist, P. Hendriks, R. Paulussen en J. Trienekens. Quint II. "Kwaliteit van softwareproducten", Kluwer Bedrijfswetenschappen, Deventer, ISBN 90-267-2430-6.
[59] Página de QUINT2. The Extended ISO Model of Software Quality. http://www.serc.nl/quint-book/, obtenida el 7-abr.-08.
[60] PENICHET, Victor, LOZANO, Maria, PIATINI, Mario. "Using WQM for classifying usability metrics". Proceedings of the IADIS International Conference. 2006; ISBN.: 972-8924-19-4, Murcia, Spain; 05 oct. 2006.
[61] VAZQUEZ, Pedro, MORENO, María, GARCIA, Francisco. Métricas Orientadas a Objetos. Technical Report, DPTOIA-IT-2001-002. Departamento de Informática y Automática, Universidad de Salamanca. 2001.
[62] MEYER, Bertrand. Object – Oriented Software Construction. Prentice-Hall ISE Inc. 2ª Edición, 1991
[63] GAFFNEY, John Jr. *"Metrics in software quality assurance"*. Portal ACM. 1991
[64] ALVARO, Alexandre, SANTANA DE ALMEIDA, Eduardo, ROMERO DE LEMOS MEIRA, Silvio. *"Quality attributes for a component quality model".* Disponible en: http://research.microsoft.com/~cszypers/events/WCOP2005/09%20-%20Alvaro.pdf. Visitada: 8 abril 2008
[65] Software Quality Assurance Group. Software Quality Assurance Handbook. KCP-613-4118. Technical communications, Kansas City Division. 1990
[66] McCONNELL, Steven. Code Complete: A Practical Handbook of Software Construction. Microsoft Press. 2ª Edición, 2004
[67] CHIMDAMBER, Shilam, KEMERER, Chis. "*Metrics suite for oriented design*". IEEE Transactions on software engineering. Vol. 20, Nº 6. Junio 1994.
[68] GONZALEZ, Heidi. Las Métricas de Software y su Uso en la Región. Escuela de Ingeniería. Departamento de Ingeniería en Sistemas Computacionales. 2001.
[69] PRITCHETT, William IV. "*An object-oriented metrics suite for Ada 95*". Portal ACM. 2001
[70] BASILI, Victor, BRIAND, Lionel, MELO, Wacélio. "A validation of object-oriented design metrics as quality indicators". Technical Report, Univ. of Maryland, Dep. of Computer Science, College Park, MD, 20742 USA. 1995
[71] EL-WAKIL, Mohamed, EL-BASTAWISI, Ali, BOSHRA, Mokhtar, and FAHMY, Ali. "Object-Oriented Design Quality Models – A Survey and Comparison". 2nd International Conference on Informatics and Systems, March 2004.
[72] LI, Wei, HENRY, Sally, "*Object-Oriented Metrics that Predict Maintainability*", Journal of Systems and Software, Vol. 23 no.2, pp.111-122, 1993
[73] LINDROOS, Jana. "*Code and design metrics for object-oriented systems*". Seminar on Quality Models for Software Engineering. Department of Computer Science, UNIVERSITY OF HELSINKI. 2004
[74] ANDERSSON, Magnus, VESTERGREN, Patrick. Object-Oriented Design Quality Metrics. Information Technology, Computing Science Department. Uppsala University. 2004.
[75] BAR, Holger, et. al. The FAMOOS Object-Oriented Reengineering Handbook, Disponible en: http://www.iam.unibe.ch/~famoos/handbook/. Consultado: 3 de septiembre de 2009.

[76] LAIRD, Linda, BRENNAN, Carol. Software Measurement and Estimation: A Practical Approach. Wiley - IEEE Computer Society Press. 2006
[77] FENTON, Norman, PFLEEGER, Lawrence. Software Metrics. PWS Publishing Co. 2ª Edición, 1996
[78] OJHA, Neeraj, McGREGOR, Jhon. "Object-oriented metrics for early system characterization: A crc card-based approach". Technical Report TR 94-107, Dept. of Computer Science, Clemson University, 1994.
[79] ABREU, Fernando, MELO, Walcélio, "*Evaluating the Impact of Object-Oriented Design on Software Quality*". Proceedings of the 3rd International Symposium on Software Metrics: From Measurement to Empirical Results. (METRICS'96). IEEE, Berlin, Germany, March 1996.
[80] CHAMPEAUX, Dennis, HORNER, Simon, MILLER, Granville. "*OO Process and Metrics for Effort Estimation*". ACM Press, Vol. 6, pg 138-142.1995.
[81] BERNÁRDEZ, B, DURÁN, A, TORO, M. "*Una Propuesta Para la Verificación de Requisitos Basada en Métricas*". Revista de Procesos y Métricas de las Tecnologías de la Información (RPM) VOL. 1, Nº 2, 2004.
[82] RIEL, Arthur. Object-Oriented Design Heuristics. Addison Wesley. 1ª Edición, 1996.
[83] NASA. Software Formal Inspections Guidebook. Disponible en: http://satc.gsfc.nasa.gov. Visitada: 16 febrero de 2009
[84] OGC, itSMF INTERNATIONAL, itSMF ESPAÑA Y EXIN. DICCIONARIO INGLÉS-ESPAÑOL DE TÉRMINOS DE GESTIÓN DEL SERVICIO DE TECNOLOGÍAS DE LA INFORMACIÓN (TI). Disponible en: http://www.cyta.com.ar/biblioteca/bddoc/bdlibros/itil_glosario.htm. Visitada: 16 octubre de 2009
[85] MANSO, Esperanza, CRESPO, Yania, DOLADO José Javier. "*Caracterización de Productos Software con Métricas no Redundantes*". JISBD 2002: 177-188
[86] LINDELL, Jonas, HAGGLUND, Mats. "*Maintainability metrics for object oriented systems*". Disponible en: http://web.abo.fi/~kaisa/HL.pdf. Visitada: 9 abril 2008
[87] LORENZ, Mark, KIDD, Jeff. Object-Oriented Software Metrics: A Practical Guide. Prentice-Hall. 1994.
[88] LANZA, Michele, MARINESCU, Radu. Object-Oriented Metrics in Practice. Springer. 2006.
[89] CHOQUE, Guillermo. Ingeniería del Software: principios y conceptos. 2002.
[90] GENERO, Marcela, PIATTINI, Mario, ALARCOS, Coral. "*A Survey of Metrics for UML Class Diagrams*". JOURNAL OF OBJECT TECHNOLOGY. Vol. 4, No. 9, November-December 2005.
[91] MYERS, Glendfor. Software Reliability: Principles and Practices. New York, NY: John Wiley & Sons; 1ª Edición. 1976.
[92] McCABE, Thomas. "*A Complexity Measure*". IEEE Transactions on Software Engineering, December 1976.
[93] MILLS, Everald. Software Metrics. SEI Curriculum Module SEI-CM-12-1. Carnegie Mellon University. 1988.
[94] McCABE, Thomas, WATSON, Arthur. "*Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric*". NIST Special Publication 500-235. 1996
[95] BELLIN David, TYAGI, Manish, TYLER, Maurice. "*Object- Oriented Metrics: an overview*". IBM Centre for Advanced Studies Conference. 1994

[96] PANAS, Thomas, LINCKE, Rudiger, LUNDBERG, Jonas, LOWE, elf. "*A Qualitative Evaluation of a Software Development and Re-Engineering Project*". IEEE Computer Society. 2005
[97] BIEMAN, James, KANG, Byung-Kyoo, "*Cohesion and reuse in an object-oriented system*". ACM press. 1995
[98] DURAN, Amador, RUIZ, Antonio, TORO, Miguel, "*Implementing Automatic Quality Verification of Requirements with XML and XSLT*". 2008. Disponible en: http://en.scientificcommons.org/42307809. Visitado: 8 abril de 2009.
[99] AJITHA et. al. Software Testing Guide Book. Part I: Fundamentals of Software Testing. Software Testing Research Lab. Disponible en: http://www.SofTReL.org. Visitado: 13 mayo 2009
[100] LANGE, Christian. Empirical Investigations in Software Architecture Completeness. Technische Universiteit Eidhoven. 2003.
[101] Latin Bussiness Ltda. Disponible en: http://www.trabajofreelance.com/do/perfil-latinbs. Visitado: 17 noviembre 2009
[102] SITI Ltda. Disponible en: http://www.siticol.com/. Visitado: 17 noviembre 2009
[103] Input Technologies Ltda. Disponible en: www.input.com.co/. Visitado: 17 noviembre 2009
[104] Seratic Ltda. Disponible en: http://www.seratic.com/. Visitado: 17 noviembre 2009