Universidad de las Ciencias Informáticas

Facultad 9

SIMETSE – SIstema de METricas para evaluar el Software Educativo.

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

AUTORES: Yudisleidys Peña Lemus

Yuniersy Hernández Díaz

TUTOR: Ing. Febe Ángel Ciudad Ricardo

CO-TUTOR: Ing. Sandy González Ruiz

ASESOR: MSc. Elianis Cepero Fadraga

Ciudad de la Habana, Julio, 6, 2007

Año del 48 Aniversario Triunfo de la Revolución

“Cuando puedas medir lo que estás diciendo y expresarlo en números, sabrás algo acerca de eso; pero cuando no puedes medirlo, cuando no puedes expresarlo en números, tus conocimientos serán escasos y no satisfactorios”

Lord Kelvin

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Yuniersy Hernández Díaz

_____
Firma del Autor

Yudisleidys Peña Lemus

_____
Firma del Autor

Febe Angel Ciudad Ricardo

_____
Firma del Tutor

OPINIÓN DEL TUTOR DEL TRABAJO DE DIPLOMA

Título: SIMETSE-Sistema de METricas para evaluar el Software Educativo.

Autores: Yudisleidys Peña Lemus.

Yuniersy Hernández Díaz.

El tutor del presente Trabajo de Diploma considera que durante su ejecución el estudiante mostró las cualidades que a continuación se detallan.

<Aquí el tutor debe expresar cualitativamente su opinión y medir (usando la escala: muy alta, alta, adecuada) entre otras las cualidades siguientes:

- Independencia

- Originalidad

- Creatividad

- Laboriosidad

- Responsabilidad>

<Además, debe evaluar la calidad científico-técnica del trabajo realizado (resultados y documento) y expresar su opinión sobre el valor de los resultados obtenidos (aplicación y beneficios) >

Por todo lo anteriormente expresado considero que el estudiante está apto para ejercer como Ingeniero Informático; y propongo que se le otorgue al Trabajo de Diploma la calificación de **<nota>**. **<Además, si considera que los resultados poseen valor para ser publicados, debe expresarlo también>**

Febe Angel Ciudad Ricardo.

_____ _____

Firma Fecha

AGRADECIMIENTOS

Primeramente agradecerle a nuestro Comandante en Jefe Fidel Castro Ruz por haber tenido la hermosa idea de crear la Universidad de las Ciencias Informáticas, que es una excelente casa de estudios.

A nuestros padres, que siempre están presentes para que nuestras derrotas no sean destructivas sino constructivas, además de sembrar en nosotras la semilla de los principios y valores, sin los cuales no vale la pena vivir.

A nuestro tutor el Ingeniero Febe Angel Ciudad Ricardo por sus consejos, su orientación, su ayuda y su paciencia.

A todos los profesores de nuestra universidad ya que directa o indirectamente siempre nos dan lo mejor de cada uno, para que seamos mejores hombres y mujeres en el futuro.

A nuestros amigos porque gracias a ellos sabemos lo que es la amistad verdadera, valor importante en nuestras vidas. Gracias por estar siempre, por aconsejarnos, regañarnos, compartir risas y llantos en todo este tiempo.

A nuestros amigos de la universidad por permitinos conocerlos y ser parte de su vida. Por ayudarnos y estar con nosotras a lo largo de la carrera.

A todos nuestros familiares y vecinos que cada dia estan presente y nos dan su ayuda incondicional.
A todas aquellas personas que ya no están con nosotras y que nos ayudaron tanto.

DEDICATORIA

A mi mamá Olga y a mi abuela María
por sus esfuerzos incansables por darme lo mejor que me pudieron dar: Educación,
Valores, Tenacidad, Regaños y todo su Amor.
A mis amistades y familiares,
por estar siempre presentes, por su amor y apoyo incondicional.
Yuniersy.

A mis padres: Mercedes y Jose luis por confiar en mí, por quererme tanto.
A mi hermano, que es mi sol, por apoyarme en tantos y tantos momentos.
A mis abuelos por estar siempre conmigo, por su amor.
A Maidiel, por llegar a mi vida, gracias por tu paciencia.
Yudisleidys.

RESUMEN

Los métodos de producción de software están evolucionando desde formas artesanales a la producción industrial en gran escala. A nivel mundial la realización de software educativo ha evolucionado grandemente por lo tanto existe la necesidad de garantizar software educativo de calidad. Cuando se hace referencia a la calidad del software educativo, se requiere de un producto que satisfaga tanto las expectativas del docente como de los alumnos, a un menor costo, libre de errores y que cumpla especificaciones instruccionales y tecnológicas. Esta necesidad conlleva a construir un modelo que especifique esta calidad, enfocándolo no sólo como un producto sino considerando también el proceso para construirlo.

Teniendo en cuenta el creciente uso del software educativo proponemos en nuestro trabajo como objetivo principal el planteamiento de un sistema de métricas que ayuden a evaluar la calidad del software educativo en la Universidad de las Ciencias Informáticas (UCI). Para cumplir este objetivo se adaptará el Modelo Sistémico de Calidad (MOSCA), a partir de tres de sus categorías: Funcionalidad, Usabilidad y Fiabilidad. Esta tesis presenta un primer capítulo que se enfocará en los temas que le dan fundamentación teórica a los conocimientos tratados, un capítulo 2 donde se expone la situación actual de la UCI con respecto a las métricas y a la calidad de los software educativos que se elabora en la universidad, y en un tercer capítulo se plantea el sistema de métricas: SIMETSE, propuesto por las autoras que plantea un conjunto de métricas adaptadas al software educativo de la UCI, se pueden encontrar la descripción exhaustiva de la solución propuesta para darle respuesta al problema científico planteado.

PALABRAS CLAVE

Software educativo, calidad del software educativo, sistema métricas.

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS.

ÍNDICE DE TABLAS

INTRODUCCIÓN

El software se ha convertido en un tema crítico en la sociedad moderna mundial. Todos parecen necesitar mejores software en menos tiempo y a menor costo. Los métodos intuitivos de desarrollo de software que se usan actualmente son, básicamente, aquellos que los propios individuos artesanalmente siguen, los cuales sólo servirán mientras la sociedad pueda tolerar la falta de predicción que ellos acarrean.

Durante muchos años, el software se desarrollaba y era utilizado por la misma persona u organización. Como todos los programas se construían de forma personalizada, los desarrolladores de este software doméstico dictaban los costos, planificación y calidad. Hoy, todo esto ha cambiado. Actualmente, cuando se va a desarrollar un software intervienen muchas personas, pero el cliente es el que tiene el problema en su empresa y desea solucionarlo. Para esto existe un analista de sistema quien es el encargado de hacerle llegar todos los requerimientos y necesidades que tiene el cliente a los programadores, personas encargadas de realizar lo que es la codificación y diseño del sistema para después, probar e instalar el software. Es así como intervienen varias personas ya que una sola persona no podría determinar todo lo necesario.

Actualmente el desarrollo del software se ha incrementado, gran parte del mismo ha sido realizado de forma desorganizada y poco documentada. Considerando el aumento exponencial que sufrirá en los próximos años, surge la necesidad de lograr una metodología disciplinada para su desarrollo, mediante los métodos, procedimientos y herramientas, que provee la ingeniería de software para construir programas educativos de calidad.

La calidad del software es un concepto que ha ido variando con los años y existe una gran variedad de formas de concebirla. Ha pasado a ser definida por grandes estudiosos del tema, tales como:

Kaoru Ishikawa, define la calidad como: “Desarrollar, diseñar, manufacturar y mantener un producto de calidad que sea el más económico, el más útil y siempre satisfactorio para el consumidor”. (K.ISHIKAWA 1998)

Pressman la define como “concordancia del software con los requisitos explícitamente establecidos, con los estándares de desarrollo expresamente fijados y con los requisitos implícitos, no establecidos formalmente que desea el usuario”. (PRESSMAN 1993)

De forma general, la calidad del software es una necesidad para las empresas. La calidad es un conjunto de características y propiedades que representan una ventaja estratégica permitiendo actuar sobre las tareas menos eficientes, mejorándolas.

La calidad del software es medible y varía de un sistema a otro o de un programa a otro, puede medirse después de elaborado el producto, pero esto puede resultar muy costoso, por lo que es imprescindible tener en cuenta tanto la obtención de la calidad como su control durante todas las etapas del ciclo de vida del software, es decir, debe correr paralela desde la planificación del producto hasta la fase de producción del mismo.

La industria del software, a diferencia de otras industrias, tiene muy poco tiempo de existir. Lo que ha llamado la atención del mercado hacia ella han sido dos factores esenciales: la velocidad con que ha crecido y su alcance. Con el tiempo los costos se redujeron y el software se convirtió en un negocio rentable, debido a esto muchas personas empezaron a desarrollar software y ahí nacieron las grandes empresas de software. Esto trajo consigo un problema: al haber tantos desarrolladores en distintos países y para distintas aplicaciones empezó a haber diversidad de estilos, así como la calidad del producto final variaba mucho entre producto y producto. En este marco se hizo necesario un estándar que permitiera a los consumidores de software decidir si el producto que estaban recibiendo era de calidad y si cumplía ciertos requisitos de funcionalidad.

Debido a esto han sido muchos los departamentos de universidades, organismos de normalización o investigación nacionales o internacionales, sociedades de profesionales, departamentos de defensa, departamentos de calidad y procesos de empresas los que han ido generando normas y estándares. Este compendio considera como entidades de mayor reconocimiento internacional, por sus trabajos y esfuerzos realizados para la normalización, y reconocimiento de la Ingeniería del software a: ISO, IEEE y SEI.

La Organización Internacional para la Estandarización o International Organization for Standarization (ISO), es una organización internacional no gubernamental que produce normas internacionales industriales y comerciales. Dichas normas se conocen como normas ISO.

El IEEE corresponde a las siglas de The Institute of Electrical and Electronics Engineers, el Instituto de Ingenieros Eléctricos y Electrónicos, una asociación técnico-profesional mundial dedicada a la estandarización, entre otras cosas. Es la mayor asociación internacional formada por profesionales de las

nuevas tecnologías, como ingenieros eléctricos, ingenieros en electrónica, ingenieros en sistemas e ingenieros en telecomunicación.

Cuando los requisitos de calidad de software se definen, se listan las características de calidad de software o sub-características que contribuyen a los requisitos de calidad. Entonces, se especifican las métricas, que estas proveen la información necesaria para la toma de decisiones técnicas. Las métricas no son estándares.

Una métrica, según Fenton es “una asignación de un valor a un atributo de una entidad de software, ya sea un producto o un proceso”. (FENTON 1991)

Las métricas son un buen medio para entender, controlar, predecir y probar el desarrollo software y los proyectos de mantenimiento. Pueden ser utilizadas para que los profesionales e investigadores puedan tomar las mejores decisiones.

La Universidad de las Ciencias Informáticas (UCI) es uno de los centros surgidos a raíz de la Batalla de Ideas. La UCI no es solo una entidad educativa, sino también productiva, que realiza múltiples proyectos destinados a la informatización de la sociedad cubana, la comercialización de software, así como a otros programas solidarios que ejecuta nuestro país a nivel mundial. A pesar de que la institución se dedica a producir software educativo, no existe conocimiento de las métricas del software por lo que no se aplican en la evaluación del software educativo. En el desarrollo de los productos no se refleja una organización y planificación adecuada, los productos son revisados en la etapa final, lo cual trae consigo que el producto no sea entregado en la fecha establecida, por lo cual existe una pérdida de tiempo y recursos tanto material como humano. (CUE. 2007)

Todo lo anterior conlleva a buscar una solución al siguiente problema: Insuficiente aplicación de métricas para la evaluación de la calidad del software educativo en la UCI.

El objeto de estudio que persigue este trabajo es el proceso de evaluación de la calidad del desarrollo del software educativo en la UCI, tomando como campo de acción el proceso de desarrollo de software educativo en la Universidad de la Ciencias Informáticas, para proporcionar la visión interna necesaria y poder así crear proyectos efectivos.

Basado en esa idea se define como objetivo general del presente trabajo: La definición de un sistema de métricas para evaluar el software educativo en la UCI (SIMETSE).

Con este trabajo se pretende motivar y aumentar el uso de las métricas apoyándose en la siguiente hipótesis: SIMETSE contribuirá a la calidad del software educativo en la UCI.

Para alcanzar los objetivos se realizan varias tareas:

- Estudiar el estado del arte de las normas internacionales utilizadas para evaluar la calidad del software.
- Investigar la calidad que presenta el software educativo que se produce en la UCI.
- Verificar si se hace un uso efectivo de las métricas para la evaluación del software educativo en la UCI.
- Incorporar a la solución propuesta elementos distintivos de otros elementos, haciendo un estudio mediante un recorrido por algunas de las métricas del software.
- Potencializar la solución en MOSCA.
- Definir SIMETSE para la evaluación del software educativo en la UCI.

Para lograr todo lo antes planteado este trabajo se enfocará en la técnica de muestreo probabilística: Muestreo aleatorio simple. Se definió este muestreo ya que se escogió para la realización de las encuestas a los líderes de proyectos de software educativo. Se definió la población de forma simple, en un listado con los elementos que la integraban. La técnica de muestreo empleada nos da una información de calidad a partir de las muestras utilizadas.

Para la realización de las tareas se emplearon los métodos siguientes:

Métodos Teóricos. Permiten estudiar las características del objeto de investigación que no son observables directamente.

- Histórico-Lógico. Los métodos históricos analizan la trayectoria completa del fenómeno, revela las etapas principales de su desenvolvimiento y las conexiones históricas fundamentales. Los métodos lógicos se basan en el estudio histórico del fenómeno, ponen de manifiesto la lógica interna de su desarrollo, de su teoría y hallan el conocimiento mas profundo de su esencia.
- Análisis y Síntesis. El análisis permite la división mental del fenómeno en sus múltiples relaciones y componentes para facilitar su estudio y la síntesis establece mentalmente la unión

entre las partes previamente analizadas, posibilita descubrir sus características generales y las relaciones esenciales entre ellas.

Métodos Empíricos. Describen y explican las características fenomenológicas del objeto, representan un nivel de la investigación cuyo contenido procede de la experiencia y es sometido a cierta elaboración racional.

- Entrevista. La entrevista es una conversación planificada entre el investigador y el entrevistado para obtener información. Su uso constituye un medio para el conocimiento cualitativo de los fenómenos.
- Encuesta. Se realiza cuando la información que se realiza puede ser obtenida a partir de la respuesta que una persona o varias puedan dar a un cuestionario preelaborado, y las mismas están dispuestas a colaborar con la investigación.

Este trabajo de diploma estará distribuido en tres capítulos:

- Capítulo 1: Se abordan los conceptos asociados al dominio del problema: normas y métricas. Se muestra una investigación de como se evalúan las métricas en los diferentes niveles organizacionales (PSP, TSP, CMM). También se plantea cuáles son los factores que influyen en el software educativo y diferentes métodos de cómo evaluar los software educativos; enfocándose en la evaluación interna y externa, la norma ISO/IEC 9126 y el Modelo Sistémico de Calidad (MOSCA). El capítulo concluye con una descripción de la situación actual de la evaluación del software educativo en la UCI.

- Capítulo 2: Se describe detalladamente la situación de las métricas en la evaluación del software educativo en la UCI, se plantea donde se encuentran las principales dificultades a la hora de evaluar los software. Se explica la importancia de las métricas y se hace una crítica en cuanto al trabajo de estas en la UCI.

- Capitulo 3: Se define SIMETSE, sistema de métricas para evaluar y mejorar la producción de software educativo en la UCI.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

Las tecnologías de información a lo largo del tiempo han hecho cada vez más fáciles las tareas que realizan las personas, esto ha traído muchos beneficios debido a que se pueden hacer más eficazmente ahorrando mucho tiempo. Pero en estos momentos, la pregunta es ¿Qué tan eficiente son estas tareas? ¿Qué tanto podemos confiar en un programa más que en una persona?

Desde la identificación del fenómeno "crisis del software", han sido muchas las organizaciones que han abordado, con mayor o menor rigor, el análisis de problemas en el desarrollo de sistemas de software. Sus trabajos se han encaminado a la localización de las causas; y a la exposición en textos didácticos, normativos o estándares de procesos o prácticas necesarias para abordar el desarrollo, mantenimiento y operación con las mayores garantías de éxito.(Victoria-Gasteiz. 2006-2007)

Este capítulo te brinda diferentes conceptos asociados al dominio del problema. Se realiza un recorrido comenzando con las normas y entidades que se dedican a la estandarización. Se lleva a cabo una descripción exhaustiva sobre las métricas de software, los distintos niveles organizacionales y culminamos este capítulo con las diferentes formas de evaluar los software educativos para luego aterrizar sobre la situación de los software educativos en la UCI.

1.2 ¿Qué se entiende por norma o estándar?

Según ISO, un estándar es "un conjunto de acuerdos documentados que contienen especificaciones técnicas u otros criterios precisos para ser usados constantemente, como reglas, lineamientos o definiciones de características. Todo esto con la finalidad de asegurar que los materiales, productos, procesos y servicios son óptimos para su propósito". (Álvarez 2004)

Un estándar es el documento aprobado por consenso por un organismo reconocido, que proporciona reglas, pautas y/o características para uso común, con el objeto de obtener un óptimo nivel de resultados en un contexto dado. (ISO/IEC 1991)

Una norma (tecnología) o estándar es toda actividad documentada que norma el comportamiento de un grupo de personas. Los estándares nos dan los medios para que todos los procesos se realicen siempre de la misma forma, mientras nos surjan ideas para mejorarlos. Son nuestra guía para la productividad y la

calidad. Es decir, las normas son un modelo, un patrón, ejemplo o criterio a seguir. Cada norma tiene un campo de validez que define la aplicación. Por esta razón, muchos productos están sujetos a muchas normas. Una norma tiene valor de regla y tiene por finalidad definir las características que deben poseer un objeto y los productos que han de tener una compatibilidad para ser usados a nivel internacional.

La estandarización o norma se basa en el trabajo conjunto de todas las partes involucradas: productores, profesionales, usuarios, administración pública, etc. Además recoge los deseos, las propuestas de todas las instituciones relevantes como son los fabricantes, las asociaciones de consumidores, los juristas, los centros de investigación, las entidades de certificación e inspección.

Las normas cubren todos los aspectos técnicos relacionados con la información, su producción y su gestión. Son coherentes y consistentes. Han sido desarrollados por comités técnicos bajo supervisión de un organismo especializado.

1.3 Entidades para la estandarización.

Han sido muchos los departamentos de universidades, organismos de normalización o investigación nacionales o internacionales, sociedades de profesionales, departamentos de defensa, departamentos de calidad y procesos de empresas los que han ido generando normas y estándares. Se considera como entidades de mayor reconocimiento internacional, por sus trabajos y esfuerzos realizados para la normalización, y reconocimiento de la Ingeniería del software a: ISO, IEEE- Computer Society y SEI. (Victoria-Gasteiz. 2006-2007)

1.3.1 IEEE

El IEEE corresponde a las siglas de The Institute of Electrical and Electronics Engineers, el Instituto de Ingenieros Eléctricos y Electrónicos, una asociación técnico-profesional mundial dedicada a la estandarización, entre otras cosas. Es la mayor asociación internacional formada por profesionales de las nuevas tecnologías, como ingenieros eléctricos, ingenieros en electrónica, ingenieros en sistemas e ingenieros en telecomunicación. Surgió en 1963 con la fusión del AIEE (Instituto Americano de Ingenieros Eléctricos) y el Instituto de Ingenieros de Radio (IRE).

El IEEE es una autoridad líder y de máximo prestigio en las áreas técnicas derivadas de la eléctrica original: desde ingeniería computacional, tecnologías biomédica y aeroespacial, hasta las áreas de energía eléctrica, control, telecomunicaciones y electrónica de consumo, entre otras. Según el mismo

IEEE, su trabajo es promover la creatividad, el desarrollo y la integración, compartir y aplicar los avances en las tecnologías de la información, electrónica y ciencias en general para beneficio de la humanidad y de los mismos profesionales. Su misión es preservar, investigar y promover la información de las tecnologías eléctricas y electrónicas. Su finalidad es avanzar en la teoría, práctica y aplicación de las tecnologías de la información. Realiza conferencias, publicaciones, cursos de formación, y desarrolla estándares.

1.3.2 SEI

El Departamento de Defensa de los Estados Unidos le preocupaba el establecimiento de métricas para identificar a los contratistas potenciales referentes a esta rama, por esta razón crearon el Software Engineering Institute (SEI). El SEI es un centro de investigación y desarrollo sostenido por el Departamento de Defensa y operado por la Universidad Carnegie Mellon. El propósito principal de este instituto es ayudar a las empresas a buscar mejoras en cuanto a la ingeniería de software basadas en métricas.

Los trabajos y aportaciones realizadas por el SEI a la Ingeniería del Software son referente mundial de primer orden, siendo la aportación más significativa los modelos de madurez de las capacidades: CMM y CMMI; que en sus casi 15 años de implantación efectiva en entornos de producción de software han demostrado su efectividad en las dos finalidades que cubren: como marco de referencia para mejora de procesos, y como criterio de evaluación para determinar la madurez, y por tanto fiabilidad de resultados previsibles de una organización de software.

1.3.3 ISO

La Organización Internacional para la Estandarización o International Organization for Standarization (ISO), es fundada en 1947. Es una organización internacional no gubernamental que produce normas internacionales industriales y comerciales. Dichas normas se conocen como normas ISO.

La misión de la ISO es promover el desarrollo de la estandarización y las actividades con ella relacionadas en el mundo, con el objetivo de facilitar el intercambio de servicios y bienes, y para promover la cooperación en la esfera de lo intelectual, científico, tecnológico y económico. Todos los trabajos realizados por la ISO resultan en acuerdos internacionales los cuales son publicados como Estándares

Internacionales. La finalidad principal de las normas ISO es orientar, coordinar, simplificar y unificar los usos para conseguir menores costes y efectividad.

1.4 Métricas de software.

Aunque los términos medida, medición y métricas se utilizan a menudo indistintamente, existe gran confusión a la hora de referirnos a ellos. Dentro del contexto de la ingeniería del software, una medida "proporciona una indicación cuantitativa de extensión, cantidad, dimensiones, capacidad y tamaño de algunos atributos de un proceso o producto"(Pressman 2002). La medición "es el proceso por el cual los números o símbolos son asignados a atributos o entidades en el mundo real tal como son descritos de acuerdo a reglas claramente definidas"(FENTON 1991), también es definida como el acto de determinar una medida.

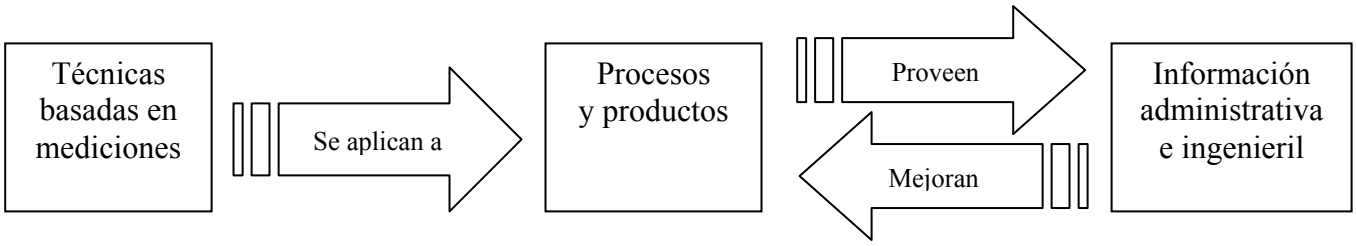
Las métricas se pueden definir como:

"La continua aplicación de técnicas basadas en la medición al proceso de desarrollo de software y a sus productos para proveer información administrativa, significativa y oportuna, junto con el uso de esas técnicas para mejorar el proceso y sus productos". (Westfall 1995)

El IEEE "Standard Glosary of Software Engering Terms" define métrica como "una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado". (IEEE 1993)

"Las métricas son un buen medio para entender, monitorizar, controlar, predecir y probar el desarrollo software y los proyectos de mantenimiento".(L.Briand 1996)

Las métricas de software proveen la información necesaria para la toma de decisiones técnicas. En la figura 1 se ilustra una extensión de esta definición para incluir los servicios relacionados al software como la respuesta a los resultados del cliente.



Las métricas son la maduración de una disciplina, van a ayudar a la (1) evaluación de los modelos de análisis y de diseño, (2) proporcionarán una indicación de la complejidad de diseños procedimentales y de código fuente, y (3) ayudarán en el diseño de pruebas más efectivas. Es por eso que propone un proceso de medición, el cual se puede caracterizar por cinco actividades:

- Formulación: La obtención de medidas y métricas del software apropiadas para la representación de software en cuestión.
- Colección: El mecanismo empleado para acumular datos necesarios para obtener las métricas formuladas.
- Análisis: El cálculo de las métricas y la aplicación de herramientas matemáticas.
- Interpretación: La evaluación de los resultados de las métricas en un esfuerzo por conseguir una visión interna de la calidad de la representación.
- Realimentación: Recomendaciones obtenidas de la interpretación de métricas técnicas trasmitidas al equipo de software.

1.5 Diferentes enfoques de métricas.

Muchos investigadores han intentado desarrollar una sola métrica que proporcione una medida completa de la complejidad del software. Aunque se han propuesto docenas de métricas o medidas, no todas proporcionan suficiente soporte práctico para su desarrollo. Algunas demandan mediciones que son demasiado complejas, otras son tan esotéricas que pocos profesionales tienen la esperanza de entenderlas, y otras violan las nociones básicas intuitivas de lo que realmente es el software de alta calidad. Es por eso que se han definido una serie de atributos que deben acompañar a las métricas efectivas de software, por lo tanto la métrica obtenida y las medidas que conducen a ello deben cumplir con las siguientes características fundamentales:

- Simple y fácil de calcular: debería ser relativamente fácil de aprender a obtener la métrica y su cálculo no obligará a un esfuerzo o a una cantidad de tiempo inusuales.
- Empírica e intuitivamente persuasiva: la métrica debería satisfacer las nociones intuitivas del ingeniero de software sobre el atributo del producto en evaluación (por ejemplo: una métrica que mide la cohesión de un módulo debería aumentar su valor a medida que crece el nivel de cohesión).

- Consistente en el empleo de unidades y tamaños: el cálculo matemático de la métrica debería utilizar medidas que no lleven a extrañas combinaciones de unidades. Por ejemplo, multiplicando el número de personas de un equipo por las variables del lenguaje de programación en el programa resulta una sospechosa mezcla de unidades que no son intuitivamente concluyentes.

- Independiente del lenguaje de programación: las métricas deberían apoyarse en el modelo de análisis, modelo de diseño o en la propia estructura del programa. No deben depender de la sintaxis o semántica del lenguaje de programación.

- Un mecanismo eficaz para la realimentación de calidad: la métrica debería suministrar al desarrollador de software información que le lleve a un producto final de superior calidad.

- Consistentes y objetivas: La métrica debería siempre producir resultados sin ambigüedad. Un tercer equipo debería ser capaz de obtener el mismo valor de métrica usando la misma información del software.

No obstante que la mayoría de las métricas de software compensan las características anteriores, algunas de las métricas usualmente empleadas no cumplen una o dos características. Un ejemplo es el punto funcional (PF), en donde se consigue argumentar que el atributo es consistente y objetivo pero falla por que un equipo ajeno independiente puede no ser capaz de conseguir el mismo valor de PF que otro equipo que emplee la misma información del software. En tal caso nos asalta la siguiente pregunta ¿Debería entonces rechazar la medida PF? La respuesta es no. Este es un ejemplo en donde el uso de las métricas depende de los factores individuales del desarrollador o desarrolladores del software.

1.6 Clasificación de las métricas del software.

Existen muchas formas de clasificar las métricas del software, distintas unas de otras según los autores. Las métricas del software pueden ser (1) Directas o Indirectas, (2) Primarias o Secundarias, (3) Internas o Externas, (4) Públicas o Privadas, (5) Simples o Complejas, (6) De Proceso, de Producto o de Proyecto, (7) Primitivas o Calculadas, etc.

Pressman clasifica el campo de las métricas en 6 categorías o grupos de métricas distintos:

- Métricas técnicas. Se centran en las características del software por ejemplo: la complejidad lógica, el grado de modularidad. Mide la estructura del sistema, el cómo esta hecho, es decir, están centradas en las características del software más que en su proceso de desarrollo.

- Métricas de calidad. Proporcionan una indicación de cómo se ajusta el software a los requisitos implícitos y explícitos del cliente. Es decir cómo voy a medir para que mi sistema se adapte a los requisitos que me pide el cliente.

- Métricas de productividad. Referidas al rendimiento del proceso de desarrollo como función del esfuerzo aplicado. Se centran en el rendimiento del proceso de la ingeniería del software. Es decir que tan productivo va a ser el software que voy a diseñar.

- Métricas orientadas al tamaño. Es para saber en que tiempo voy a terminar el software y cuantas personas voy a necesitar. Son medidas directas al software y al proceso por el cual se desarrolla.

- Métricas orientadas a la función. Son medidas indirectas del software y del proceso por el cual se desarrolla. Las métricas orientadas a la función se centran en la funcionalidad o utilidad del programa.

- Métricas orientadas a la persona. Proporcionan medidas e información sobre la forma que la gente desarrolla el software de computadoras y sobre todo el punto de vista humano de la efectividad de las herramientas y métodos. Son las medidas que voy a hacer de mi personal que hará el sistema.

1.7 PSP, TSP, CMM

Para darle solución a muchos de los problemas o errores que se pueden ocasionar después de finalizado el producto se debe tener en cuenta métricas muy elevadas de calidad así como modelos muy bien definidos para garantizar que el software es de excelente calidad.

En un principio se utilizaba un modelo llamado CMM (Capability Maturity Model) que fue desarrollado por el instituto de ingeniería de software (SEI). Este modelo es muy utilizado en la actualidad, y de este hay dos procesos que se deben de tomar en cuenta hoy en día en cualquier desarrollo de software, Personal Software Process (PSP) y Team Software Process (TSP).

1.7.1 PSP: Proceso Personal de Software.

“El proceso de PSP consiste en un conjunto de métodos, formatos y reglas que muestran a los ingenieros como plantear y administrar su trabajo. PSP esta diseñado para su uso con cualquier lenguaje de

programación o método de diseño y puede ser usado para la mayor parte de aspectos de trabajo incluyendo requerimientos de trabajo, correr pruebas, definir procesos y reparar defectos". (Hayes 1997)

PSP se concentra en las prácticas de trabajo de los ingenieros en una forma individual. El principio detrás de PSP es ése, sirve para producir software de calidad, cada ingeniero debe trabajar en la necesidad de realizar trabajo de calidad. Los métodos de PSP son presentados en una serie de siete versiones de procesos. Estas versiones son denominadas como PSP0 a PSP3. Cada versión tiene un mismo conjunto de *logs*, formularios, *scripts*, y *standards.* Para observar las diferentes versiones del PSP ver anexo 1.

1.7.1.1 Métricas en el PSP

Con datos de tamaño, tiempo y defectos, existen muchas formas de medir, evaluar, y manejar la calidad de un programa. PSP provee una serie de mediciones de calidad que ayudan a los desarrolladores a examinar la calidad de sus programas desde varias perspectivas. Como ninguna medición por sí sola puede indicar adecuadamente la calidad de un programa, el panorama que provee la utilización de todas estas mediciones es generalmente un indicador confiable de calidad.

Las principales mediciones de calidad son:

- Densidad de defectos
- Índice de revisión
- Índices de tiempo de desarrollo
- Índices de defectos
- Rendimiento
- Defectos por hora
- Efectividad de remoción de defectos
- Evaluación del índice de fallas

Mediante el Plan del Proyecto se define el trabajo y como se hará el mismo. El Plan del Proyecto cuando esta adecuadamente documentado, es un punto de referencia para comparar con el rendimiento real. A través del mismo se puede estimar el tamaño del producto, el tiempo en que estimamos hacer el trabajo, los defectos introducidos en cada fase, los defectos que eliminamos en cada fase, etc. Es recomendable que los planes del proyecto se analicen cuando el proyecto esté terminado y que se use como punto de

partida para el análisis de proyectos similares, aunque esto no exime que los planes se obtengan cada vez que el usuario los desee y les sirva entonces como guía de referencia del estado del proyecto. Para una mejor comprensión de cómo se trabaja con el Plan del Proyecto véanse el anexo 2, en el cual se muestra paso a paso cómo completarlo.

El problema más grande que se presenta con el PSP es que para los programadores es difícil tratar de visualizar todas las tareas y etapas del desarrollo. Es por eso que en muchas ocasiones se necesita de otros programadores, pero al estar más de uno no se puede utilizar el proceso de software personal, este problema se soluciona formando un equipo de trabajo y llevando a cabo el desarrollo por medio de TSP (Team Software Process).

1.7.2 TSP: Proceso de Software del Equipo.

El proceso del equipo de software (TSP) es un proceso que está basado en el modelo CMM. TSP está diseñado para ayudar a controlar, administrar y mejorar la forma en que trabaja un equipo de software. Tiene como objetivo construir y sostener un entorno de equipos que trabajen de manera cohesionada. Estos equipos deben atender las necesidades del negocio, mejorar la calidad, y reducir los costes y tiempos.

El propósito del TSP, es ayudar a los miembros de un equipo de software a formar el equipo, establecer sus relaciones de trabajo, determinar los roles de los miembros, ponerse de acuerdo en las metas del equipo y las metas y funciones de cada uno de sus miembros. Por otra parte ayuda a los miembros del equipo a ser cooperativos y a lograr un trabajo personal disciplinado, de forma tal que planifique y le dé seguimiento a su propio trabajo y sobre la base de producir productos con calidad; siempre orientados a alcanzar las metas del equipo. (Humphrey. 2000.)

Debido a que la mayoría de los proyectos no se deja a un solo individuo sino que se deja a un equipo de trabajo el TSP es un proceso de desarrollo muy utilizado por las diferentes instituciones y compañías de todo el mundo.

Sin embargo el utilizar TSP tiene requisitos, entre ellos se encuentra:

- Contar con personal suficientemente capacitado.
- Que el personal tenga la capacidad de trabajar en equipo.
- Debe haber convivencia en el equipo para facilitar la comunicación.

- Tienen que realizarse roles de trabajo y cada integrante debe estar en el área en la que mejor se desarrolla para que sea mas eficiente.

El TSP es sin duda alguna un proceso infalible para tener software de la mejor calidad, los programadores solo tienen que adecuarse a el y utilizarlo de manera apropiada para obtener todos los beneficios que a ellos les trae. Al igual que PSP, está estructurado por formularios, guías y procedimientos para desarrollar software, por lo que utiliza métricas para evaluar a sus programadores, las cuales se pueden observar en el anexo 3.

1.7.3 CMM: Modelo de Madurez de la Capacidad.

Según Paulk, "describe los principios y practicas relacionadas al proceso de madurez de software e intenta ayudar a las organizaciones de software a mejorar la madurez de sus procesos de software en términos de un camino evolutivo, desde procesos caóticos hasta procesos maduros y disciplinados ".

El Modelo de Madurez de la Capacidad o CMM (Capability Maturity Model), es un método para definir y gestionar los procesos a realizar por una organización. Los procesos para desarrollar software a gran escala, pueden ser muy grandes y complejos. Por esta razón fue que el (SEI) de la Universidad Carnegie-Mellon desarrolló un ambiente de trabajo (framework) de madurez de procesos de software. Este framework es una forma ordenada para las organizaciones de determinar las capacidades de sus procesos actuales y establecer prioridades en su mejora. Se hace a través del establecimiento de 5 niveles de prioridad progresivos logrando procesos de capacidades más maduros. Por cada nivel se han definido los principios elementales o KPAs (Key Process Areas) que proveen las metas y ejemplifican las prácticas a llevar a cabo. CMM ha sido revisado y refinado por varios especialistas y representan el mejor juicio actual y el método más efectivo para conseguir los objetivos de cada nivel de madurez. Establece cinco niveles ascendentes de madurez: 1) Inicial, 2) Repetible, 3) Definido, 4) Administrado, 5) Optimizado. (Ver anexo 4).

CMM a partir del nivel dos (Repetible) comienza a utilizar las métricas ya que se considera que en los próximos niveles será muy tardado llevar a cabo una transición. CMM plantea métricas para sus desarrolladores, para el proceso y producto. Los productos son evaluados mediante métricas en las diferentes etapas del desarrollo en el anexo 5 se pueden ver algunos ejemplos de métricas planteadas para evaluar las revisiones.

1.7.4 Relación entre PSP, TSP y CMM

- CMM/CMMI, TSP y PSP pueden usarse de forma y combinada para mejorar las capacidades de toda la organización.
- PSP forma ingenieros de equipos establecidos con TSP en la mayoría de las prácticas genéricas de CMM/CMMI. TSP por si solo, aunque sea aplicado a todos los equipos de desarrollo, no cubre todas las prácticas de cada área de proceso de CMM/CMMI, razón de más para que sea utilizado de forma complementaria a este modelo, no de forma aislada.
- Debido a que las actividades de medición y análisis de los resultados son fundamentales en PSP y en TSP, su utilización durante la aplicación de CMM/CMMI en una organización, permite acelerar el progreso y aumentar el nivel de capacidad de la empresa en un tiempo menor que sin su uso.

1.8 Métricas del proceso.

Las métricas del proceso de software se utilizan para propósitos estratégicos. Por ejemplo: actividades definidas, como el desarrollo de un sistema de software desde los requerimientos hasta la liberación del usuario o la inspección de una parte del código. Las métricas de proceso también se extraen midiendo las características de tareas específicas de la ingeniería de software y obteniendo como resultados medidas de errores detectados antes de la entrega del software, defectos detectados e informados por los usuarios finales, productos de trabajo entregados, el esfuerzo humano y tiempo consumido, ajuste con la planificación y otras medidas. Por ejemplo: (FENTON 1991)

Ejemplo 1: La fórmula siguiente se podría usar como una medida indirecta para un proceso, como una prueba formal:

Costo/ Número de errores encontrados

Esta fórmula nos da el promedio entre el costo y el número de errores encontrados durante el proceso, en donde el costo serían los atributos de procesos externos los cuales se cree que son importantes tales como: controlabilidad, observabilidad y estabilidad

Ejemplo 2: La fórmula siguiente se podría usar como la base para una medida de la estabilidad del proceso del diseño durante un período específico de tiempo.

Estabilidad de diseño = Número total de metodologías de diseño

Número de procesos del diseño

Las métricas de proceso se recopilan de todos los proyectos y durante un largo periodo de tiempo. El objetivo es proporcionar indicadores, estos indicadores permiten a una organización de ingeniería del software tener una visión profunda de la eficacia de un proceso ya existente; por ejemplo: el paradigma, las tareas de ingeniería del software, los productos del trabajo e hitos. También permiten que los administradores evalúen lo que funciona y lo que no.

Grady (R.G.Grady 1992.) argumenta que existen usos "privados y públicos" para diferentes tipos de datos del proceso. Los ingenieros de software podrán sentirse no familiarizados con la utilización de métricas recopiladas de una base particular, estos datos deberían ser privados para el individuo y servir sólo como un indicador de ese individuo, algunos ejemplos podrían ser los índices de defectos, los índices de defectos por módulo, errores encontrados durante el desarrollo, ya que sabemos que los datos privados de proceso pueden servir como referencia importante para mejorar el trabajo individual del ingeniero de software.

Mayhauser menciona que algunas métricas de proceso son privadas para el equipo del proyecto de software, pero públicas para todos los miembros del equipo. Entre los ejemplos se incluyen los defectos informados de funciones importantes del software (que un grupo de profesionales han desarrollado), errores encontrados durante revisiones técnicas formales y líneas de código o puntos de función por módulo y función. El equipo revisa los datos para detectar los indicadores que pueden mejorar el rendimiento del equipo.

Las métricas del proceso del software pueden proporcionar beneficios, sin embargo, al igual que todas las métricas, estas pueden usarse mal, originando más problemas de los que pueden solucionar. Grady (R.G.Grady 1992.) sugiere una etiqueta de métricas del software adecuada para gestores al tiempo que instituyen un programa de métricas de proceso:

- Utilice el sentido común y una sensibilidad organizativa a1 interpretar datos de métricas.
- Proporcione una retroalimentación regular para particulares y equipos que hayan trabajado en la recopilación de medidas y métricas.
- No utilice métricas para evaluar a particulares.

- Trabaje con profesionales y equipos para establecer objetivos claros y métricas que se vayan a utilizar para alcanzarlos.
- No utilice nunca métricas que amenacen a particulares o equipos.
- Los datos de métricas que indican un área de problemas no se deberían considerar negativos. Estos datos son meramente un indicador de mejora de proceso.
- No se obsesione con una sola métrica y excluya otras métricas importantes.

A medida que una organización está más a gusto con la recopilación y utilización de métricas de proceso, la derivación de indicadores simples abre el camino hacia un enfoque más riguroso, llamado Mejora de Estadística del Proceso del Software (MEPS). Pressman menciona que MEPS utiliza el análisis de fallas del software para recopilar: (Pressman 1998)

- Información de errores y defectos encontrados en el desarrollo y utilizar una aplicación de sistema o producto. El análisis de fallos funciona de la siguiente manera:
- Todos los errores y defectos se categorizan por origen (p. ej.: defectos en la especificación, en la lógica, no acorde con los estándares).
- Se registra tanto el costo de corregir cada error como el del defecto.
- El número de errores y de defectos de cada categoría se cuentan y se ordenan en orden descendente.
- Se calcula el costo global de errores y defectos de cada categoría que producen el costo más alto para la organización.
- Los datos resultantes se analizan para detectar las categorías que producen el costo más alto para la organización.

Se desarrolla planes para modificar el proceso con el intento de eliminar (o reducir la frecuencia de apariciones) la clase de errores y defectos que sean más costosos.

1.9 Factores o parámetros que influyen en el software educativo.

Los factores o parámetros que influyen en el desarrollo o producción de software educativo son de índole educativa, tecnológica y económica (A.Galvis 2000; Pressman 2002), tal como se muestra en la figura 1.

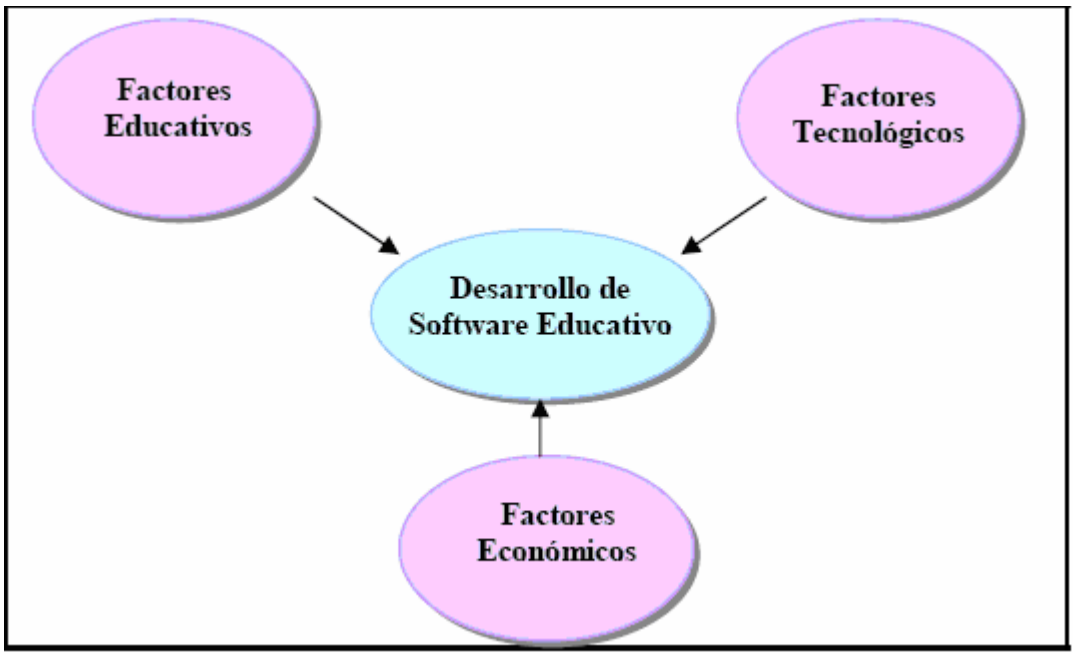


Figura 1: Factores y parámetros que influyen en la producción de software educativo.

- Factores educativos. Además de conocer lo que significa software educativo y definir el tipo de software que se va a desarrollar, se deben conocer también los factores educativos que intervienen en el proceso de desarrollo de software educativo como lo son las teorías de aprendizaje, las teorías de diseño instruccional, los estilos de aprendizaje en la Web, la elección y desarrollo de los contenidos, los objetivos de la enseñanza, la metáfora utilizada, las estrategias de enseñanza, la forma de presentación más adecuadas, la evaluación de la enseñanza y las pautas para la evaluación de software educativo desde el punto de vista del educador.

- Factores tecnológicos. Los parámetros que se relacionan con la tecnología y el desarrollo de software educativo, estarán relacionados con el estudio, desarrollo y aplicación de máquinas, equipos y técnicas para la manufactura y los procesos de producción de software educativo (Pressman 2002). A la hora de diseñar un software educativo se deben tomar en cuenta los parámetros siguientes: el diseño de la interfaz, el modelo de desarrollo y producción de software y la calidad de software. El recurso humano relacionado con el área tecnológica está formado por especialistas o expertos en diseño de software, tales como dibujantes, diseñadores gráficos e ilustradores, expertos en programación, profesionales dedicados a ingeniería de software o

desarrollo de software, expertos en digitalización de sonido y audio como también profesionales expertos en calidad de software.

- Factores económicos: Estos parámetros están relacionados con el financiamiento del proyecto o inversión económica que se debe realizar para la elaboración del software educativo. Surge la pregunta ¿cuánto cuesta producir un software educativo? La respuesta depende de quién lo produzca y quién lo financia. Existen varios tipos de financiamiento:
  - De la inversión del Estado o gobierno.
  - De la inversión del sector privado.
  - De la inversión mixta, del estado y del sector privado.

Si lo financia el estado a través de entidades públicas tales como universidades o institutos de investigación, el costo del proyecto sólo refleja parte de los costos debido a que sólo se financia el costo de algunos equipos, parte de materiales y suministros, parte del personal, parte de los servicios y viajes. Los costos que no están incluidos directamente, el estado los está financiando indirectamente al cubrir los costos totales de funcionamiento de estas instituciones. Por esta razón es difícil calcular cuánto le cuesta al estado producir software educativo.

Además de escoger el modelo de desarrollo de software adecuado para los propósitos de elaborar un software educativo, se debe tomar en cuenta los parámetros de calidad de software. Cuando se habla de calidad de software educativo, se requiere un producto que satisfaga tanto las expectativas de los docentes como de los usuarios, a un menor costo, libre de defectos y cumpliendo con ciertas especificaciones. (Pressman 2002)

1.10 Evaluación del software educativo. Diferentes enfoques.

En las últimas décadas se han elaborado muchas propuestas sobre software educativo, gran parte de estas son de índole cualitativa o necesitan adaptarse a medidas estándares de evaluación de software. Si bien varían en cuanto a contenido y estilo, todas ellas tienen un objetivo común, que es ayudar al docente a elegir y valorar un programa adecuado. En este trabajo se muestra algunos de los métodos que podemos utilizar para evaluar el software educativo. Este trabajo se enfocará en la evaluación interna y externa, la ISO/IEC 9126 y MOSCA. A continuación se presentan los aspectos más importantes acerca de estos diferentes modos de evaluación.

1.10.1 Evaluación del software educativo mediante la evaluación interna y externa.

La evaluación de los programas educativos es un proceso que consiste en la determinación del grado de adecuación de dichos programas al contexto educativo. Básicamente, se realizan las evaluaciones interna y externa del software, a fin de detectar los problemas que generarán cambios en el producto, lo antes posible, a fin de reducir costos y esfuerzos posteriores. Estas evaluaciones consideran las eventuales modificaciones sugeridas por el equipo de desarrollo y por los usuarios finales, teniéndose en cuenta a docentes y alumnos en el contexto de aprendizaje. Cuando un producto del tipo comercial educativo, llega al docente, significa que ha superado las etapas de evaluaciones interna y externa. Además para obtener el grado de eficacia y de eficiencia del producto se deberá realizar una evaluación en el contexto de uso.

La evaluación interna

La evaluación interna del software estará a cargo de los miembros del equipo de desarrollo (que se realiza al obtener la versión alfa).

Bork (BORK 1986), denomina a la evaluación interna como formativa, o sea la evaluación del proceso, como aquella realizada generalmente por los desarrolladores.

La evaluación externa

Mediante la evaluación externa se obtendrán sugerencias de los alumnos, quiénes serán en definitiva los usuarios del software y de los docentes que lo utilizarán como material didáctico. Durante este tipo de pruebas, se encuentran errores imprevistos no detectados y se verifica el cumplimiento de los programas con los objetivos educativos que se han considerado en el diseño.

Bork (BORK 1986) la denomina evaluación sumativa y es la evaluación del producto final que generalmente la realizan equipos distintos a los desarrolladores.

Como resultado de ambas evaluaciones, se obtendrá la primera versión del programa con su respectivo manual de usuario, conteniendo todos los aspectos que se consideren indispensables para el uso docente, con detalles técnicos, y del entorno pedagógico y didáctico en el que se desarrolló el programa.

Instrumentos de evaluación

Los instrumentos de evaluación más usados, son los cuestionarios de valoración, donde las respuestas a estos cuestionarios son valoradas entre 0 y 5, por ejemplo, siendo el resultado el grado de conformidad del usuario con las afirmaciones propuestas.

Los instrumentos de evaluación, en forma de planillas se deben confeccionar con inclusión de preguntas del tipo cerradas, abiertas, y casillas de verificación, permitiendo al usuario final la descripción de aspectos problemáticos y particulares del programa que no hayan sido tenidos en cuenta durante la confección del instrumento. Se deberá tener en cuenta al redactar los cuestionarios la utilización de un vocabulario adecuado, sin ambigüedades y claro para los destinatarios previstos en cada caso en particular.

Como cada propuesta de evaluación de software es particular, se deben analizar con cuidado las diferentes propuestas de evaluación de medios didácticos y en particular de software educativo, teniéndoselas sólo como una "guía" que luego se deberá "readaptar" a cada contexto educativo particular.

1.10.2 Evaluación del software educativo según MOSCA.

Según las normas ISO/IEC 9126 (ISO/IEC 1991), surge la necesidad de la disponibilidad de un instrumento de medidas estándares de calidad para la evaluación de software educativo, que sea de utilidad tanto para los desarrolladores de software educativo como para los interesados en adquirir software comercial (por ejemplo, educadores e instituciones educativas). Se propone entonces un modelo de evaluación de software educativo bajo un enfoque sistémico de calidad, basado en El Modelo Sistémico de Calidad de Software (MOSCA).(L.Mendoza, Pérez et al. 2001)

En cuanto al producto, este modelo toma como base las 6 características de calidad del estándar internacional ISO/IEC 9126, En cuanto al proceso, utiliza las 5 características de calidad del estándar internacional ISO/IEC 15504. MOSCA consta de cuatro niveles:

- Nivel 0: Dimensiones
- Nivel 1: Categorías
- Nivel 2: Características
- Nivel 3: Métricas

1.10.3 Evaluación del software educativo según ISO/IEC 9126.

La norma ISO/IEC 9126 posee cuatro partes:

- Parte 1: Modelo de Calidad.
- Parte 2: Métricas Externas.

- Parte 3: Métricas Internas.
- Parte 4: Métricas de Calidad en Uso.

El estándar identifica seis atributos clave de calidad:

- Funcionalidad: El grado en que el software satisface las necesidades. Indicada por los siguientes subatributos: Adecuación, Exactitud, Interopabilidad, Seguridad de acceso y Cumplimiento funcional.
- Fiabilidad: Cantidad de tiempo que el software está disponible para su uso. Estará referido por los siguientes subatributos: Madurez, Tolerancia a fallos, Facilidad de recuperación y Cumplimiento de la fiabilidad.
- Usabilidad: Grado en que el software es fácil de usar. Viene reflejado por los siguientes subatributos: Capacidad para ser entendido, Capacidad para ser aprendido, Capacidad para ser operado, Capacidad de atracción y Cumplimiento de la usabilidad.
- Eficiencia: Grado en que el software hace óptimo el uso de los recursos del sistema. Viene reflejado por los siguientes subatributos: Comportamiento frente al tiempo, Uso de recursos y Cumplimiento de la eficiencia.
- Mantenibilidad: La facilidad con que una modificación puede ser realizada. Está indicada por los siguientes subatributos: Capacidad para ser analizado, Capacidad para ser cambiado, Estabilidad, Capacidad para ser probado y Cumplimiento de la mantenibilidad.
- Portabilidad: La facilidad con que el software puede ser llevado de un entorno a otro. Está referido por los siguientes subatributos: Adaptabilidad, Instalabilidad, Coexistencia, Capacidad para reemplazar y Cumplimiento de la portabilidad.

La norma define un modelo de calida basado en dos partes bien identificadas: la calidad interna y externa y la calidad de uso.

1.10.3.1 Métricas internas.

Las métricas internas pueden aplicarse a un producto de software no ejecutable (como una especificación o código fuente) durante el diseño y la programación. Las métricas internas miden los atributos interiores o indican los atributos externos por medio del análisis de las propiedades estáticas del producto de

software intermedio o final. En el desarrollo de un producto de software los productos intermedios deben evaluarse usando las métricas internas que miden las propiedades intrínsecas, incluyendo las que pueden derivarse de un comportamiento simulado. El propósito primario de estas métricas internas es asegurar el logro de la calidad externa y la calidad durante el uso requerido. Las métricas internas constituyen una ventaja para los usuarios, evaluadores, verificadores, y diseñadores, pues le permiten evaluar la calidad de producto de software y atender los aspectos de la calidad desde las etapas más tempranas.

1.10.3.2 Métricas externas.

Las métricas externas usan valores de un producto del software derivadas de las mediciones del comportamiento del sistema del que es parte, mediante el ensayo, la operación y observación del software o sistema ejecutable. Antes de adquirir o usar un producto del software, el mismo debe evaluarse usando métricas basadas en objetivos comerciales relacionados con el uso, explotación y gestión del producto en un ambiente organizativo y técnico especificado. Éstas son las métricas externas primarias. Las métricas externas constituyen una ventaja para los usuarios, evaluadores, verificadores, y diseñadores pues le permiten evaluar la calidad del producto de software durante el ensayo y la operación.

1.10.3.3 Métricas de Calidad en uso.

La evaluación de la calidad en el uso valida la calidad del producto de software en situaciones específicos de las tareas del usuario. Las métricas de la calidad en el uso miden hasta que punto un producto satisface las necesidades de usuarios específicos para lograr las metas especificadas con la efectividad, productividad, seguridad y satisfacción en un contexto determinado de uso.

- Efectividad: Capacidad del producto software para permitir a los usuarios alcanzar objetivos especificados con exactitud y completitud, en un contexto de uso especificado.
- Productividad: Capacidad del producto software para permitir a los usuarios gastar una cantidad adecuada de recursos con relación a la efectividad alcanzada, en un contexto de uso especificado.
- Seguridad física: Capacidad del producto software para alcanzar niveles aceptables del riesgo de hacer daño a personas, al negocio, al software, a las propiedades o al medio ambiente en un contexto de uso especificado.

- Satisfacción: Capacidad del producto software para satisfacer a los usuarios en un contexto de uso especificado.

1.11 Situación actual en la UCI con respecto a la evaluación del software educativo.

Hoy día es cada vez más frecuente la consideración de métricas de software, es por eso que sé están implantando en la actualidad, llevando consigo puntos débiles (aumento de esfuerzo, etc) y fuertes (alta calidad, reusabilidad, madurez, etc) que están experimentado los ingenieros y administradores de software. El uso de éstas se ha adoptado con éxito en el amplio mercado de desarrollo de software introduciendo reconocimientos y consideraciones por parte de administradores y usuarios, y estableciendo la necesidad de un enfoque más disciplinado y de una alta calidad. Así muchos particulares y compañías desarrolladoras de software, están reconociendo la importancia del uso de las métricas, aunque de igual modo siguen sin conocer el alcance de madurez y calidad del producto final y la disciplina de ingeniería madura que llega a alcanzar con la aplicación de los distintos métodos y técnicas y la interpretación de los resultados que proyecta el uso de las métricas.

La producción de software en Cuba se ha ido incrementando positivamente y el surgimiento de la Universidad de las Ciencias Informáticas (UCI) ha sido un factor determinante para el desarrollo de la producción de software. La UCI no solo es una entidad educativa, en ella también se lleva a cabo la producción de software; interviniendo en la creación de los mismos tanto estudiantes como profesores de la universidad. La universidad está dividida en facultades, cada una esta identificada con un perfil. Para darle cumplimiento a estos perfiles se desarrollan un conjunto de proyectos productivos. Uno de los perfiles definidos y que además determina una fuerte línea en la producción en la universidad es el Software Educativo.

Hoy día ya existe un Sistema Metodológico para guiar el proceso de producción en la universidad, creado sobre las bases de las dificultades que se fueron presentando en los proyectos anteriores. En este Sistema Metodológico para el control del estado de avance de los proyectos se estableció un esquema orientado a dos objetivos específicos: (1) Auditorias, (2) Control de avance. El plan de las auditorias consiste en un cronograma de revisiones que se ejecuta por la especialista de control de los proyectos, apoyándose de una lista de chequeo que permite mediante métricas de software evaluar el estado del proyecto.

Según las encuestas y entrevistas realizadas principalmente a líderes de proyecto se puede afirmar que existe un desconocimiento total sobre el tema de las métricas por lo que podemos llegar a la conclusión de que este sistema no se esta cumpliendo en la universidad. Los proyectos que se producen en la universidad son entregados al departamento de calidad para ser evaluados en su fase final, realizándose solo las pruebas de caja negra y caja blanca. En numerosas ocasiones, la fecha de entrega al departamento de calidad coincide con la fecha de entrega al cliente, lo cual trae consigo que los proyectos sean entregados de forma retrasada.

Las encuestas y entrevistas arrojaron que la dificultad principal que presentan los productos educativos elaborados en la universidad, se debe fundamentalmente en gran medida a la no utilización y seguimiento de metodologías y estándares o normas de calidad establecidos a nivel internacional para el desarrollo de productos software. Los productos se caracterizan por tener poca planificación o ninguna, la estimación de los tiempos de desarrollo es irreal, la evaluación de los productos se realiza en la etapa final de su desarrollo y existe ausencia de métricas para evaluar la calidad de los productos.

1.12. Conclusiones

Luego de haber analizado los diferentes enfoques de la evaluación del software, la importancia que presentan las métricas en la evaluación de estos, se arriba a las siguientes conclusiones:

- En la UCI hay un desconocimiento practicamente general de las métricas de software.
- Se hace necesario un sistema de métricas para evaluar el software educativo en la UCI, con ellas se podría evaluar la calidad del software que se produce, lo cual trae consigo que cuando el software es entregado al usuario final, este presente la calidad requerida.
- Para evaluar el software educativo que se produce en la UCI el método más apropiado es el Modelo Sistémico de Calidad (MOSCA).

CAPÍTULO 2: SITUACIÓN ACTUAL DE LAS MÉTRICAS DE SOFTWARE.

2.1 Introducción.

Este capítulo tiene como objetivo definir como se comporta la Universidad de las Ciencias Informáticas (UCI) con respecto a la utilización de las métricas de software y cuál es la calidad de los proyectos educativos que actualmente se elaboran en la universidad. Además se pretende crear una cultura de medición en la comunidad de la UCI y que mediante este trabajo de diploma puedan conocer las ventajas que presentan las métricas para evaluar la calidad de los software educativos.

Los resultados que se plasman en el presente capitulo fueron obtenidos de las encuestas y entrevistas realizadas por las autoras a los líderes de los proyectos educativos, así como al personal que de una forma u otra estaba involucrado con el proceso de desarrollo de software en la UCI.

“Las mediciones pretenden ayudar a los administradores de proyectos a tomar decisiones, no toman decisiones automáticamente”. [McGraw, 1998]

2.2 La importancia de medir.

El área de las mediciones de software, a pesar de ser una de las áreas en la ingeniería de software donde se ha investigado desde hace más de 30 años, todavía no ha sido bien comprendida. Las métricas todavía no son ampliamente aplicadas en la industria del software, a pesar de la gran cantidad de literatura sobre el tema, la cual provee una buena variedad de ejemplos para el uso de métricas de software en términos de incrementar la calidad y productividad. Por tanto, la implementación de las mediciones en una organización requiere un cambio cultural importante.

Suele haber preocupación de que las mediciones señalarán problemas en un proyecto o en una organización que no eran visibles antes de que el proceso de medición fuera implementado. Por ejemplo el análisis de las mediciones puede mostrar que los planes de desarrollo no son realistas, o que determinado programador está atrasado en sus tiempos de entrega. Estas preocupaciones son reales, y sobreponerse a ellas, requiere un entendimiento de las mediciones, así como saber como usar los resultados de las mediciones apropiadamente en todos los niveles de la organización. (McGarry, Bailey et al. 1998)

Debido a esto nos encontramos con algunos de los siguientes problemas que conllevan a que el software se haga cada vez más impreciso y difícil a la hora de medir:

- No se cuenta con un catálogo lo suficientemente completo y estructurado, con información clasificada relacionada con métricas e indicadores que sirva como base para la selección de las mismas y como apoyo al diseño del proceso de medición o evaluación
- La información disponible sobre métricas, indicadores y modelos de evaluación no está basada en criterios comunes, y en muchos casos está definida para dominios específicos, dificultando su aplicación a procesos de evaluación.
- No existe una terminología común consensuada para el ámbito de métricas e indicadores que permita una fluida comunicación de información entre distintos proyectos de evaluación y facilite la aplicación de resultados obtenidos en distintos trabajos de investigación y casos de estudio.
- No se cuenta con la satisfacción del cliente como principal criterio de calidad, y cuánto avanzamos en esta satisfacción. (Rodríguez, Nadal et al.)

2.3 Problemas en Cuba.

En estudios realizados por el Centro de Estudios de Ingeniería de Sistemas (CEIS) del ISPJAE en la Industria de Software Cubana se detectaron problemas entre los que se encuentran: (Febles 2000; Yanko Hernández 2001)

- Ausencia de la definición de los procesos de la empresa, siguiendo los principios de la ingeniería y la gestión de software
- No existe un procedimiento para llevar a cabo las actividades relacionadas con el desarrollo del software.
- No hay eventos predefinidos para hacer pedidos de cambios.
- No existe planificación del trabajo.
- No se puede dar seguimiento al progreso del proyecto (no hay definición de tareas, relaciones entre estas, cronogramas, puntos de chequeo, etc.)

- No se lleva el registro de quien hizo cada actividad relacionada con el proyecto (no se puede exigir responsabilidades, no se puede medir productividad, cantidad de errores cometidos a nivel individual).
- Al no planificar, no hay seguimiento, por lo que es imposible ver el estado del progreso del proyecto.
- No se puede comparar con proyectos anteriores y aprender de estos.
- No se pueden ver los problemas de calidad (cantidad de errores) en distintas áreas de trabajo.
- Existe dificultad para evaluar el trabajo de los especialistas al no poseer los mecanismos para detectar la magnitud del trabajo desarrollado.
- No existen procedimientos para obtener métricas que permitan planificar y controlar los procesos.
- Hay mala calidad en mucho del software que se produce en el país.
- Los proyectos están excesivamente tarde y los beneficios que pudieran obtenerse al utilizar los mejores métodos e instrumentos en las distintas etapas no se detectan en este medio indisciplinado y caótico de desarrollo
- Existe falta de comunicación efectiva entre los involucrados (usuarios, desarrolladores, administradores, clientes e investigadores)

Esencialmente, el Aseguramiento de la Calidad del Software consiste en la revisión de los productos y su documentación relacionada, para verificar su cobertura, corrección, confiabilidad y facilidad de mantenimiento. (S.KAN 1995)Y, por supuesto, incluye las métricas, que nos dan una medida para controlar la calidad del software, permitiendo detectar defectos en los proyectos de software. Las métricas son un método más para garantizar que un proyecto cumpla las especificaciones y los requisitos para el uso y desempeño deseado.

2.4 Sistema Metodológico para el desarrollo de Software Educativo en la UCI.

El sistema metodológico para el desarrollo de software educativo, realizado por un colectivo de autores de la UCI permite ilustrar la estrategia adoptada en la producción de los productos de software educativo y multimedia en la Universidad de Ciencias Informáticas para garantizar la adecuada ejecución de los

proyectos. En este sistema metodológico para el control del estado de avance de los proyectos se estableció un esquema orientado a dos objetivos específicos:

- Auditorias
- Control de avance

Para las auditorias se plantea que:

Tienen por objetivo chequear el cumplimiento de las políticas y procedimientos establecidos. Se realiza a través de listas de chequeo que integran todas las áreas definidas y establecen el nivel de exigencia de la dirección hacia las facultades en la presente etapa.

Están basadas en las políticas establecidas por la dirección que a su vez están en total concordancia con las políticas mínimas de calidad establecidas por la Dirección de Calidad de la IP. Considerando las posibilidades de cumplimiento de estas políticas por los proyectos productivos en las facultades y las especificidades de algunos de ellos.

Estos controles se efectúan por parte de la dirección de Producción de la Infraestructura productiva y por parte del grupo de calidad de la Facultad encargada de velar por la buena ejecución de los procesos de producción.

Estos mecanismos deberán perfeccionarse en adelante concibiendo un sistema de control y retroalimentación dinámica a cada uno de los proyectos a través de reportes de estado.

Para la realización se ha concebido un plan de auditorias, este plan consiste en un cronograma de revisiones que se ejecuta por la especialista de control de los proyectos, apoyándose de una lista de chequeo que permite mediante métricas de software evaluar el estado del proyecto.

2.4.1 Omisión de los principios del Sistema Metodológico.

En la entrevista realizada a Abel Ernesto Lorente, Especialista Superior que trabaja en la Dirección de Producción #2, plantea que las métricas están descritas en los procedimientos aunque los proyectos productivos no la utilicen como una vía para evaluar la calidad del software. Aunque existen y están bien descritas, como plantea Abel, no se aplican por parte del equipo de desarrollo. (Lorente 2007)

Aunque existe un sistema metodológico que plantea las auditorias y revisiones apoyándose en los datos que arrojan las métricas, este sistema no se lleva a cabo. El equipo de desarrollo al terminar el software

envía este para el departamento de calidad, donde solo se le hacen las pruebas de caja negra y caja blanca a los productos, esto nos demuestra que se le está restando importancia al término de calidad de los productos. Una mala calidad nos afecta en distintos aspectos como: económicos, morales, productivos. Un programa educativo bien diseñado, con una alta calidad, ayudará a aumentar las expectativas que tiene el pueblo cubano y las organizaciones nacionales e internacionales con la Universidad de las Ciencias Informáticas, ayudará a incrementar la calidad de la enseñanza que se le ofrece a los estudiantes y facilitará el acceso a una educación de nivel a mayor número de personas, etc.

La opinión de Abel Ernesto Lorente nos demuestra que no se le esta dando la importancia requerida a la calidad, en una escala de 1 a 5, la calidad en la UCI seria de 4 puntos, teniendo siempre en cuenta que no todos, ni la mayoría de los proyectos presenta la calidad requerida y la productividad en los proyectos educativos no es la adecuada. Abel Ernesto nos plantea: ¨La productividad en los proyectos de software educativo en la UCI es Media – Alta en algunos proyectos y en otros proyectos es Baja – Media, el por qué es simplemente un problema de responsabilidad y preparación del equipo de trabajo y las facultades que supervisan los proyectos y en otros casos las malas definiciones de los clientes.

2.5 Situación en la UCI con respecto a las métricas del software.

La Universidad de las Ciencias Informática (UCI) no esta aislada del problema de las mediciones. Hoy la mayoría de los desarrolladores de software en la UCI: líderes de proyectos, programadores, diseñadores, etc., no planean ni registran su trabajo y no miden y administran la calidad de los productos. El desconocimiento de las métricas es un factor que pesa en la evaluación del software educativo. La entrevista realizada a Rosalía Cue, Especialista en la dirección de desarrollo nos demuestra que no se emplean métricas para evaluar los productos, ella plantea: ¨No se utilizan las métricas porque el nivel de desarrollo y de organización no nos lo permite, debido a la inestabilidad existente¨.

La mayoría de los desarrolladores de software en la UCI todavía no miden, a pesar, de que las mediciones pueden ayudarlos a lograr que su trabajo cada día sea mejor y con una mayor calidad. El desconocimiento de las métricas por parte del equipo de trabajo hace que establezcan un rechazo hacia ellas. Entrevistas realizadas nos demuestran que casi todos los desarrolladores piensan que solo realizando las pruebas de caja negra y caja blanca obtienen un producto de calidad, emitiendo observaciones y sugerencias en cada uno de los errores encontrados hasta llegar al punto donde ocurre el error dando poca importancia a las métricas que nos dan una medida cuantitativa de la calidad que presenta el producto. (Hernández. 2007).

Con la utilización de las métricas se puede determinar si el producto que estamos desarrollando esta mejorando y si posee o no la calidad requerida.

En un intento por esclarecer la importancia de las métricas muchos desarrolladores oponen resistencia, no entienden para que sirven y se preguntan porque es tan necesario medir el proceso de desarrollo de software y el producto que produce. Los rigores del trabajo diario de un proyecto del software no dejan mucho tiempo para pensar en estrategias. Los líderes del proyecto de software están más preocupados por aspectos mundanos (aunque igualmente importantes): desarrollo de estimaciones significativas del proyecto, terminar el producto a tiempo y dejan a un lado la calidad del producto. Mediante el uso de la medición para establecer una línea base del proyecto, cada uno de estos asuntos se hace más fácil de manejar. La línea base sirve como un lineamiento para la estimación. Además, la recopilación de métricas de calidad permite a una organización “sintonizar” su proceso de ingeniería del software para eliminar las causas “poco vitales” de los defectos, que tienen el mayor impacto en el desarrollo del software. Las métricas proporcionan beneficios a nivel de proyecto, estratégico y técnico.

2.6 Resultados de la investigación.

La UCI no tiene personal que asuma las métricas con rigurosidad. En estudios realizados a los proyectos de software educativo que se realizan en la facultad 8 y 9 se detectaron los siguientes problemas:

- El personal disponible no presenta el nivel requerido acorde al proyecto que le asignan.
- Existe una gran desorganización en cuanto a la distribución de los proyectos.
- No existe una organización a la hora de realizar las pruebas.
- No existen planes de revisiones, se revisa solo las funcionalidades que brinda el producto o que debe brindar según el guión técnico y el guión de contenido.
- No existe una cultura de producción de software bajo parámetros de terminación y calidad, donde se actúe bajo conceptos y estudios técnicamente fundamentados por equipos multidisciplinarios y competentes dirigidos a la creación de un producto orientado a determinado mercado.
- Hay mala calidad en gran parte del software que se produce en la UCI, producto a mala organización de la documentación del producto, guiones mal elaborados, poca comunicación entre

líderes de proyectos y líderes de calidad que atienden al proyecto, entregas tardías de medias por parte de diseño, etc.

- Hay un desconocimiento en gran parte del equipo de trabajo de los proyectos, de las métricas para evaluar la calidad del software educativo.
- No se aplican las métricas para evaluar el software educativo

Es indispensable la solución de todos estos problemas lo más brevemente posible, pues la calidad del producto que se desarrolla en la UCI es clave para mejorar su competitividad, y teniendo la calidad del producto como elemento distintivo, la confianza que se depositará en la UCI va a hacer cada día mayor.

En las encuestas y entrevistas realizadas por las autoras en este año 2007, en total 11, a proyectos que desarrollan software educativo en la UCI y al personal que de una forma u otra tiene relación con la calidad del software educativo, se detectaron que a pesar de la existencia de directivas de calidad, donde se utilizan las pruebas para controlar la calidad del software educativo, y de la disponibilidad de recursos para llevarlas a cabo, no se elaboran y aprueban métricas como una alternativa para eliminar aún más los errores en los productos, existe un desconocimiento de las métricas por parte del equipo de desarrollo de los proyectos, solo se realizan pruebas al finalizar el producto y no se registran datos sobre los defectos encontrados que permitan tener una medida de la calidad de los productos y del proceso de detección de los mismos. Algunos de los resultados se muestran en las Figuras 2, 3, 4 y 5.

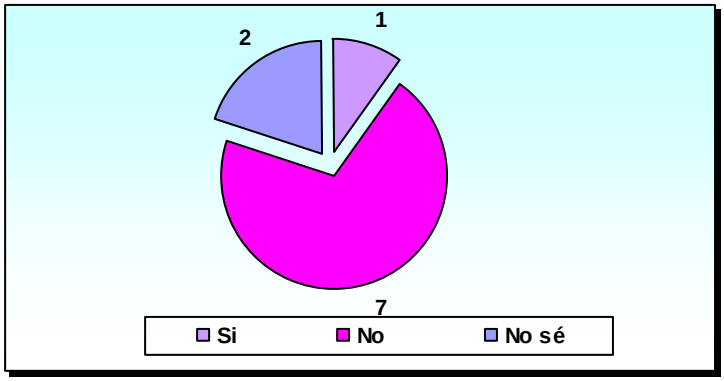


Figura 2: Utilización de las métricas para evaluar la calidad del software educativo.

La figura 2, nos demuestra que la mayoría de los proyectos y personal entrevistados no utilizan las métricas para evaluar la calidad del software educativo, el factor principal de este motivo es que existe un desconocimiento de las mismas. (Ver figura 3)

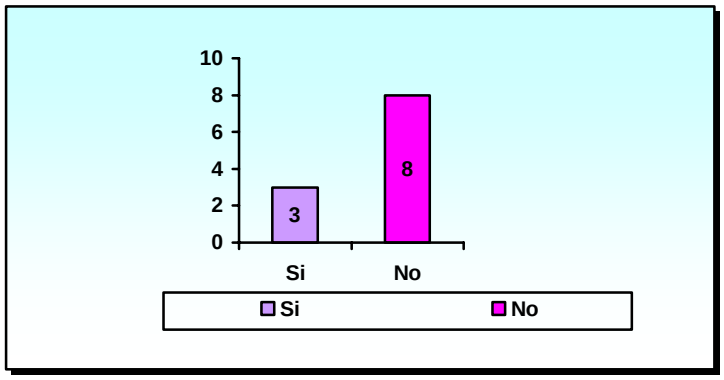


Figura 3: Conocimiento de las métricas del software.

La figura 3 demuestra que de 11 proyectos entrevistados solo 3 proyectos conocen las métricas para evaluar la calidad del software, y aunque ellos conocen acerca del tema es muy poco la información que tienen del mismo.

En la figura 4 se muestra según las encuestas, la importancia que le brindan a las métricas de software, aunque no haya un conocimiento de estas por parte del personal de la UCI, ellos plantean que las métricas presentan una gran importancia en la evaluación del software educativo.

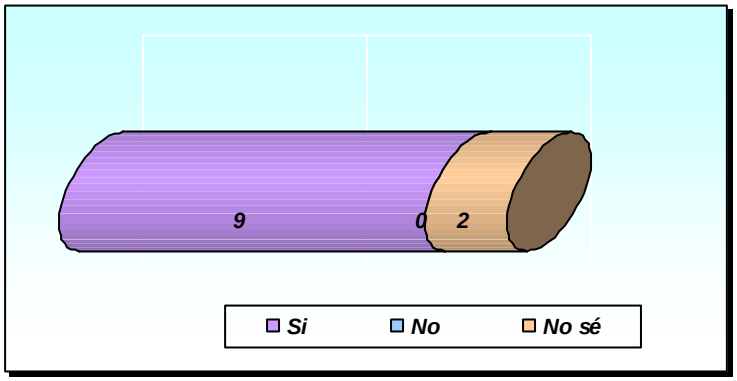


Figura 4: Importancia de las métricas de software.

En la figura 5 se demuestra que el total de entrevistados, aunque no tenga un conocimiento amplio y profundo de las métricas de software, entienden su importancia para la calidad de los productos, por lo que desean que se apliquen a los software educativos que se realizan en la UCI.

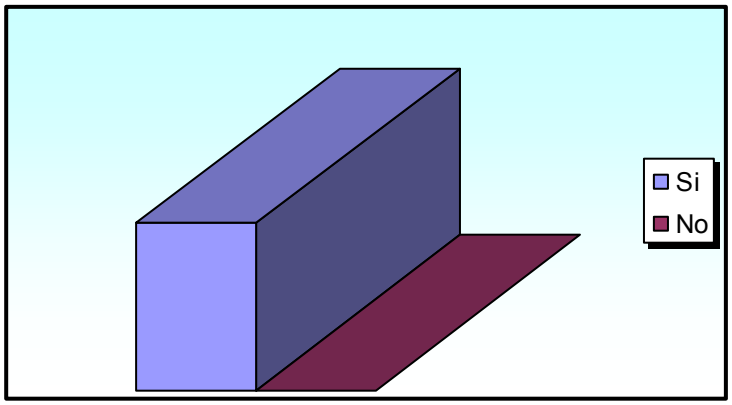


Figura 5: Aplicar las métricas para evaluar la calidad del software educativo en la UCI.

Estos resultados arrojan la importancia de dar a conocer al equipo de desarrollo de software educativo la necesidad de establecer las métricas para evaluar la calidad en los productos con el fin de mejorar la productividad en la UCI y lograr una adecuada satisfacción de los clientes tanto en el mercado nacional como en el internacional.

Establecer un adecuado Sistema de Aseguramiento de la Calidad empleando las métricas para evaluar la calidad del software educativo, y por supuesto, crear una cultura de las métricas y su importancia en los equipos de desarrollo, contribuye considerablemente a elevar la calidad en los productos resultantes del proyecto de software. Con las métricas del software se detectan errores que van desde la omisión de algún requisito, en las etapas tempranas del desarrollo del proyecto, hasta la detección de errores en el código de algún producto de software asociado con éste.

Los resultados de este estudio demuestran que aunque le proporcionen alta importancia a la calidad del producto desde el inicio del desarrollo del software no se aplican métricas para evaluar la calidad de los productos, además solo se realizan pruebas al terminar el proyecto. La calidad atiende tanto la organización como las funcionalidades que debe presentar el producto, esto es de suma importancia, puesto que estos productos al llegar a manos de los clientes deben funcionar exactamente como ellos pidieron en la contratación del producto. Si nos preocupamos por la calidad del software educativo desde la etapa de levantamiento de requisitos y hasta cada una de las fases del proyecto se contribuye a eliminar aún más lo errores al finalizar el software.

En la actualidad, en la UCI los proyectos están excesivamente tardes, se exige mayor productividad y calidad en un tiempo menor, por lo que se puede decir que las fallas en los proyectos residen principalmente en:

- No existe una planificación real: El equipo de desarrollo no aplican métodos de planificación ya que no están entrenados para esto o piensan que no es importante planificar el trabajo, esto trae que los proyectos no estén listos en el tiempo requerido.
- Una mala calidad en el software: La carencia de métricas adecuadas de calidad junto con pobres conocimientos acerca del trabajo a realizar traen consecuencias como: productos con mucho defectos, demoras, y una pobre o nula calidad que torna ineficiente el proceso de prueba.
- Personal no idóneo: En numerosas ocasiones, los proyectos son asignados a un grupo de personas que no están capacitadas para dicho proyecto, que no presentan los conocimientos que se requieren, trayendo esto consigo que haya un atraso fundamental a la hora de entregar el producto.

2.7 La motivación y la percepción: factores esenciales.

El apoyo de los líderes de proyecto es primordial para lograr una implementación exitosa de las métricas. Los líderes de proyectos junto con todo el equipo de trabajo deben tomar un interés eficaz y visible en el proceso de medición, deberán comprender la importancia de las mediciones y empezar a apoyarlas activamente. Esta conciencia de las métricas debe ir más allá de que el líder de proyecto diga que las métricas del software son una buena idea.

La motivación a nivel de proyecto es diferente. Una vez que el proyecto ha comenzado, la principal motivación para usar las métricas debe ser que permitirán a los administradores monitorear y controlar el progreso, el costo y la calidad del proyecto. (Kan 2000)Para ayudar a los equipos a vencer el miedo a las métricas, se les debe educar, decirles porque las métricas son importantes y que es lo que se va a medir, dejar claro que las métricas nunca serán usadas en contra de los miembros del equipo. Un administrador de software competente no necesita las métricas individuales para distinguir a los miembros de los equipos competentes de los remolones. (Wiegers 1999)De forma general las métricas deben ser usadas para los siguientes propósitos (Zuse 1995):

- Coleccionar datos y reportar valores de métricas en bases regulares.

- Identificar el nivel actual de cumplimiento de cada métrica.
- Tomar acciones remediales si el nivel de alguna métrica crece mal o excede los límites establecidos.
- Establecer metas de mejoramiento específico en término de las métricas.

Un aspecto importante para obtener apoyo de los equipos de trabajo en cuanto a las métricas, es asegurarse que todos comprendan que el resultado de las mediciones se usará para apoyar los objetivos de la organización, y no para evaluar el rendimiento individual. Si los individuos son castigados por pobres resultados en las mediciones, entonces el flujo de datos puede ser impedido o manipulado, trayendo como resultado una pérdida de la comprensión y la comunicación del proyecto. [McGraw, 1998].

Para lograr una motivación y percepción por parte de los equipos de desarrollo en cuanto a las métricas, es importante respetar la privacidad de los datos y de las métricas, clasificando cada elemento de dato que se coleccione en alguno de los tres niveles que propone Wiegers (Wiegers 1999).

- Individual: sólo los individuos que coleccionen los datos sobre su propio trabajo pueden acceder a sus datos, aunque se pueden combinar con los datos de otros individuos para proveer el perfil del proyecto completo.
- Equipo de proyecto: los datos son privados para los miembros del equipo, aunque se pueden combinar con los datos de otros proyectos para proveer el perfil completo de la organización/ empresa.
- Organización: los datos pueden ser compartidos por todos los miembros de la organización.

Sin embargo, el simple establecimiento de un proceso de medición no tendrá un impacto inmediato en el proyecto. Mientras se implementa el proceso de medición, los resultados de las mediciones necesitarán ser “comercializados” dentro de la organización. Deben realizarse discusiones de los resultados de las mediciones dentro del equipo de proyecto, las cuales deben enfocarse en como los resultados de las mediciones reflejan lo que está sucediendo realmente en el proyecto, y si alguna nueva cuestión identificada por el análisis de las métricas es válida o no. Es importante que el líder del proyecto esté al menos dispuesto a escuchar las “malas noticias” que resulten del análisis de las mediciones y que sepa que no todos los resultados de los análisis de las mediciones requieren acción. (Estévez 2002)

2.8 Reglas básicas para evitar el problema del factor humano.

Cuando algo es medido automáticamente asume importancia. Las personas quieren “verse bien”, por tanto quieren que las métricas los hagan “verse bien”. La mejor forma de evitar el problema del factor humano en el trabajo con las métricas es seguir algunas reglas básicas tales como las que se enuncian a continuación (Wiegers 1999):

- No mida a los individuos: El estado del arte en las métricas de software no llega hasta ese punto. Medir la productividad de los individuos nos lleva a que ellos se enfoquen solo en su propio trabajo dejando a un lado al equipo de desarrollo. Hay que llevar la dirección hacia los procesos y productos.
- Nunca usar las métricas como un “garrote”: Si utilizamos las métricas contra los individuos o contra el equipo de trabajo no obtendremos que las apoyen, y mucho menos que la utilicen.
- No ignorar los datos: No obviar los datos que arrojan las métricas que son importante en la toma de decisiones.
- Nunca usar una sola métrica: Enfocarse en una sola métrica nos puede dar que solo el atributo que es medido se beneficie por los demás que obviamos. Debe mantenerse un balance entre los atributos del costo, la calidad y los cronogramas de forma que se satisfagan todas las necesidades de los usuarios.
- Proveer retroalimentación: Proporcionando una retroalimentación regular al equipo sobre los datos que ellos ayudan a coleccionar tiene varios beneficios:
- Ayuda a mantener el foco en la necesidad de coleccionar los datos. Cuando el equipo vea que los datos están siendo realmente usados, será más probable que consideren la importancia de la colección de los datos.

Si los miembros del equipo se mantienen informados sobre los detalles específicos de cómo los datos son usados, ellos tendrán menos posibilidades de empezar a sospechar sobre su uso. Involucrando a los miembros del equipo en el análisis de los datos y en los esfuerzos por mejorar los procesos, se obtienen beneficios de conocimiento y experiencia.

La retroalimentación en problemas e integridad de la colección de datos, ayuda a educar a los miembros del equipo en la responsabilidad de coleccionar los datos. Los beneficios pueden ser datos más consistentes, exactos y oportunos.

2.9 Conclusiones.

En este capítulo se presentó una serie de factores que influyen en la calidad del software educativo, Unas de las conclusiones mas importante que se observa es la no existencia de métricas para evaluar el software educativo de la UCI, el desconocimiento que hay de estas en la comunidad universitaria. Con los gráficos mostrados de las encuestas y entrevistas realizadas, se observa la situación actual de la UCI, donde se observa que el trabajo que se presenta es de vital importancia ya que no solo trata de crear una cultura de medición en la UCI que mejorará la calidad del producto y por tanto la calidad de los procesos de desarrollo en los proyectos de software educativo.

Para lograr un sentido de pertenencia de las métricas en un programa de medición, se tiene que lograr la participación en la definición de las métricas. Las personas que trabajan con un proceso en su quehacer diario tendrán un conocimiento íntimo de ese proceso, esto les da una perspectiva valiosa de cómo el proceso puede ser mejor medido para asegurar confiabilidad y validez, y cómo interpretar mejor las mediciones resultantes para maximizar la utilidad.

CAPÍTULO 3: DEFINICIÓN DE SIMETSE – SISTEMA DE MÉTRICAS PARA EVALUAR EL SOFTWARE EDUCATIVO.

3.1. Introducción.

En este capitulo se describe la propuesta planteada “SIMETSE- Sistema de METricas para evaluar el Software Educativo”. Este sistema ayudará a evaluar la calidad del software educativo en la Universidad de las Ciencias Informáticas (UCI). La implementación de este sistema provocaría una mejora en el área de los productos educativos. Apoyado en el modelo sistémico de calidad (MOSCA), SIMETSE se basa en un conjunto de métricas que se definieron y adaptaron para el software educativo que se produce en la UCI, donde permitirá conocer el estado del producto. Estas métricas ayudarán a predecir, evaluar y caracterizar el software educativo y sobre todo ayudará a mejorar la calidad de los mismos.

3.2 Métricas del Producto.

La gestión eficaz de un proyecto de software se centra en las cuatro P: personal, producto, proceso y proyecto. El orden no es arbitrario.

Antes de poder planificar un proyecto, se deberán establecer los objetivos y el ámbito del producto, se deberían considerar soluciones alternativas e identificar las dificultades técnicas y de gestión. Sin esta información, es imposible definir unas estimaciones razonables (y exactas) del coste; una valoración efectiva del riesgo, una subdivisión realista de las tareas del proyecto o una planificación del proyecto asequible que proporcione una indicación fiable del progreso.

Para valorar la calidad del producto se pueden emplear las métricas técnicas según se va desarrollando el mismo. Las métricas técnicas para el software de computadora no son absolutas, nos proporcionan una manera sistemática de valorar la calidad basándose en un conjunto de reglas claramente definidas. Estas medidas de atributos internos del producto le proporcionan al desarrollador de software una indicación en tiempo real de la eficacia del análisis, del diseño y de la estructura del código, la efectividad de los casos de prueba, y la calidad global del software a construir.

1. Métricas del Modelo de Análisis.

El trabajo técnico en la ingeniería del software empieza con la creación del modelo de análisis. En esta fase se obtienen los requisitos y se establece el fundamento para el diseño. Por tanto, son deseables las

métricas técnicas que proporcionan una visión interna a la calidad del modelo de análisis. Dentro de las métricas del modelo de análisis encontramos (1) Métricas basadas en la función, (2) La métrica bang y (3) Métricas de la calidad de la especificación.

2. Métricas del Modelo de Diseño.

Las métricas de diseño para el software, como otras métricas del software, no son perfectas. En nuestros días diseñamos sin definir las medidas del diseño, sin determinar las métricas para varios aspectos de la calidad del diseño y no las usamos para guiar la evolución del diseño, lo cual trae consigo que el diseño sin medición se convierta en una alternativa inaceptable. Dentro de las métricas del modelo del diseño están las (1) Métricas del diseño arquitectónico o métricas de diseño de alto nivel, (2) Métricas de diseño a nivel de componentes y (3) Métricas de diseño de interfaz.

3. Métricas del Código Fuente.

La ciencia del software asigna leyes cuantitativas al desarrollo del software de computadora, usando un conjunto de medidas primitivas que pueden obtenerse una vez que se ha generado o estimado el código después de completar el diseño. Estas medidas son.

n1: el número de operadores diferentes que aparecen en el programa.

n2: el número de operandos diferentes que aparecen en el programa.

N1: el número total de veces que aparece el operador.

N2: el número total de veces que aparece el operando.

4. Métricas para Pruebas.

Aunque se ha escrito mucho sobre las métricas del software para pruebas (por ejemplo, [HET93]), la mayoría de las métricas propuestas se concentran en el proceso de prueba, no en las características técnicas de las pruebas mismas. En general, los responsables de las pruebas deben fiarse de las métricas de análisis, diseño y código para que les guíen en el diseño y ejecución de los casos de prueba. Las métricas de la prueba desembocan en las siguientes categorías:(1) métricas que ayudan a determinar el número de pruebas requeridas en los distintos niveles de la prueba; (2) métricas para cubrir la prueba de un componente dado.

A medida que se van haciendo las pruebas, tres medidas diferentes proporcionan una indicación de la compleción de las pruebas, (1) amplitud de las pruebas, (2) profundidad de las pruebas y (3) perfiles de fallos.

5. Métricas del Mantenimiento.

Todas las métricas del software presentadas anteriormente pueden usarse para el desarrollo de nuevo software y para el mantenimiento del existente. Sin embargo, se han propuesto métricas diseñadas explícitamente para actividades de mantenimiento. El estándar IEEE 982.1 - 1988 [IEE94] sugiere un índice de madurez del software (IMS) que proporciona una indicación de la estabilidad de un producto software (basada en los cambios que ocurren con cada versión del producto).

A medida que el IMS se aproxima a 1.0 el producto se empieza a estabilizar. EL IMS puede emplearse también como métrica para la planificación de las actividades de mantenimiento del software. El tiempo medio para producir una versión de un producto software puede correlacionarse con el IMS desarrollándose modelos empíricos para el mantenimiento.

3.3 Calidad del software educativo.

Galvis (A.Galvis 2000) plantea que en el campo educativo se define como software educativo aquellos programas que permiten cumplir y apoyar funciones educativas. En esta categoría se encuentran los que dan soporte al proceso de enseñanza y aprendizaje (un sistema para enseñar matemáticas, ortografía, contenidos o ciertas habilidades cognitivas).

Cuando se habla de calidad de software educativo, se requiere un producto que satisfaga tanto las expectativas de los docentes como de los usuarios, a un menor costo, libre de defectos y cumpliendo con ciertas especificaciones. (Pressman 2002)

Según Gros (Gros 2000) la calidad del software educativo está determinada no sólo por los aspectos técnicos del producto sino por el diseño pedagógico y los materiales de soporte. Este último aspecto es uno de los más problemáticos ya que existen poco programas que ofrezcan un soporte didáctico.

3.3.1 La evaluación del software educativo.

Como la cantidad de software educativo ha crecido muy rápidamente, el docente se encuentra con la necesidad cada vez mayor de evaluarlo para determinar el grado de adecuación de un programa a su propio entorno. Ellos necesitan saber cómo utilizar un programa determinado y cuándo deberían utilizarlo

para mejorar su enseñanza. Por otra parte los alumnos también deben saber cómo mediante tal o cual programa podrían mejorar sus aprendizajes.

La evaluación es para los programas educativos, la etapa más importante de todo el proceso de construcción, evaluando desde el diseño del producto y la producción del mismo, hasta el modo de uso, el tiempo y el momento de uso. La evaluación es una tarea constante a lo largo de todo el desarrollo y aún después, en el contexto de aplicación, ya que requiere también de evaluación de las estrategias cognitivas propuestas.

La evaluación de software educativo se ha centrado tradicionalmente en dos momentos:

- Durante su utilización real por los usuarios, para juzgar su eficiencia y los resultados que con él se obtienen.
- Durante el proceso de diseño y desarrollo, con el fin de corregir y perfeccionar el programa.

3.4 Modelo Sistémico de Calidad. (MOSCA)

Según Callaos y Callaos (Callaos and Callaos 1993), la calidad de los Sistemas de Software no es algo que depende de una sola característica en particular, sino que obedece al compromiso de todas sus partes. Tomando en cuenta la calidad del producto y la calidad del proceso, el LISI-USB desarrolló el Modelo Sistémico de Calidad de Software –MOSCA-, que integra el modelo de calidad del producto y el modelo de calidad del proceso de desarrollo, y está soportado por los conceptos de calidad total sistémica.

Con respecto al producto, MOSCA plantea, sobre la base de las 6 características de calidad del estándar internacional ISO/IEC 9126, un conjunto de categorías, características y métricas asociadas que miden la calidad del software convirtiendo el modelo estudiado en un instrumento que garantiza la medición ya que mide todos los aspectos imprescindible para medir directamente la calidad del producto de software. En cuanto a la perspectiva del proceso, este modelo se formuló sobre la base de las 5 características de calidad del estándar internacional ISO/IEC 15504, un conjunto de categorías, características y métricas asociadas que miden la calidad de un proceso de software con un enfoque sistémico. El modelo de calidad que soporta este enfoque se describe a continuación.

MOSCA consta de 4 niveles (ver anexo #7), los cuales son descritos seguidamente:

Nivel 0: Dimensiones. Eficiencia del proceso, Efectividad del proceso, Eficiencia del producto y Efectividad del producto son las cuatro dimensiones propuestas en el prototipo de modelo. Sólo un balance y una buena interrelación entre ellas permiten garantizar la calidad Sistémica global de una organización.

Nivel 1: Categorías. Se contemplan 11 categorías: 6 pertenecientes al producto y las otras 5 al proceso de desarrollo.

Producto: Funcionalidad (FUN), Fiabilidad (FIA), Usabilidad (USA), Eficiencia (EFI), Mantenibilidad (MAB) y Portabilidad (POR).

Proceso: Cliente-Proveedor (CUS), Ingeniería (ENG), Soporte (SUP), Gestión (MAN) y Organizacional (ORG).

Nivel 2: Características. Cada categoría tiene asociado un conjunto de características (56 asociadas al producto y 27 al proceso de desarrollo), las cuales definen las áreas claves a satisfacer para lograr, asegurar y controlar la calidad tanto en el producto como en el proceso. Entre las características asociadas a cada categoría del producto, se proponen en el modelo MOSCA, una serie de características del proceso. Esto se debe a que algunas características de la calidad del proceso, impactan directamente en las categorías del producto, al igual que ciertas características de la calidad del producto definen categorías del proceso.

Nivel 3: Métricas. La cantidad de métricas asociadas a cada una de las características que conforman MOSCA es de 587 en total.

Adicionalmente, MOSCA cuenta con un algoritmo que facilita su operacionalización y permite estimar la calidad de software. El algoritmo contempla tres fases: (1) estimación de la calidad del producto de software con un enfoque sistémico; (2) estimación de la calidad del proceso de desarrollo de software con un enfoque sistémico; y (3) integración de las mediciones de los sub-modelos de la calidad del producto y la calidad del proceso.

3.5 Adecuación de MOSCA para software educativo de la UCI.

Dado que MOSCA es un modelo de especificación de la calidad de los sistemas de software, que además permite su medición, en este trabajo se hace una adecuación del mismo para software educativo de la UCI.

Como un primer alcance, se decidió utilizar sólo la Perspectiva Producto, y de esta perspectiva sólo la Dimensión de la Efectividad del Producto; en virtud de que la necesidad más inmediata es dar a la UCI una orientación para que se evalúa el software educativo que se produce.

Para el área de software educativo, se encontró que las características y métricas indicadas en MOSCA no se adaptan completamente a este tipo de software, debido a que las métricas están diseñadas genéricamente, y por lo tanto no consideraba los aspectos pedagógicos y metodológicos del proceso de enseñanza-aprendizaje que se debe tomar en cuenta al diseñar un instrumento de evaluación. (P.Marquès 1998)

Por tal motivo se procedió a realizar las siguientes actividades para efectuar los cambios necesarios en MOSCA para adaptarlo y especificarlo en el área educativa.

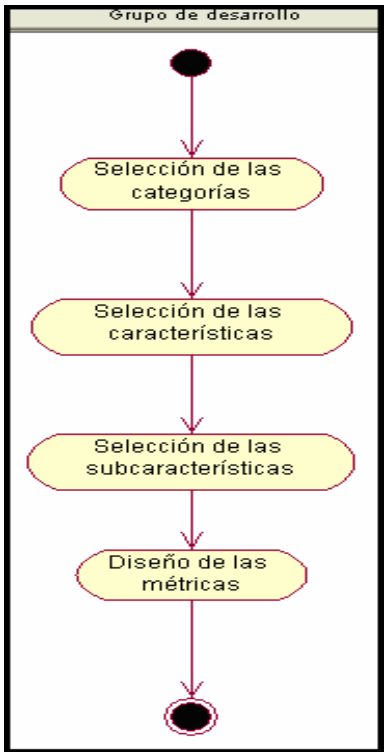


Figura 6: Diagrama de actividades. Actividades a realizar para adaptar a MOSCA al software educativo.

Explicación de las actividades.

- Selección de las categorías. MOSCA consta de seis (6) categorías, de las cuales sólo se deben utilizar 3 (tres) para la evaluación de software educativo. Debido a que la categoría de

Funcionalidad siempre debe estar presente, en esta actividad se seleccionan dos (2) categorías de las cinco (5) restantes del modelo del producto (Usabilidad, Fiabilidad, Eficiencia, Mantenibilidad y Portabilidad). Se seleccionaron Usabilidad y Fiabilidad. La Usabilidad es seleccionada debido a que un software educativo motive al aprendizaje, es fundamental que el material educativo sea atractivo y de fácil manejo, debe generar actividades interactivas que motiven y mantengan la atención, actividades que deben ser variadas y que respondan a los diversos estilos de aprendizaje. Se seleccionó Fiabilidad debido a que es importante que el producto funcione bajo las condiciones establecidas y mantenga un nivel específico de rendimiento.

- Selección y propuesta de las características y subcaracterísticas. Una vez seleccionadas las categorías que están relacionadas con la evaluación de software educativo (Funcionalidad, Usabilidad y Fiabilidad), se seleccionan las características asociadas a estas categorías en MOSCA, que estén relacionadas con el área educativa. Se decidió seleccionar ciertas características asociadas a la efectividad del producto y no a la eficiencia del producto, debido a que, al adquirir un software comercial, no se tiene acceso a los documentos que permiten aplicar las métricas correspondientes a esta dimensión, por lo que no es posible evaluarla.

- Diseño y definición de las métricas. Es necesaria una selección de métricas adicionales relacionadas con Funcionalidad, Usabilidad y Fiabilidad, que permitan adaptar MOSCA en el área de software educativo de la UCI.

El resultado de estas actividades se puede apreciar en la Tabla 1.

CATEGORÍA	CARACTERÍSTICAS	SUBCARACTERÍSTICAS
FUNCIONALIDAD (FUN) (82)	FUN.1 Ajuste a los propósitos (73)	FUN.1.1 General (3) FUN.1.2 Objetivos de aprendizaje (5) FUN.1.3 Contenidos de aprendizaje (12) FUN.1.4 Actividades de aprendizaje (11) FUN.1.5 Ejemplos (3) FUN.1.6 Motivación (10) FUN.1.7 Retroalimentación (7) FUN 1.8 Ayudas (3) FUN.1.9 Evaluación y registro de datos (10) FUN.1.10 Metodología de enseñanza (9)
	FUN.2 Precisión (5) FUN.3 Seguridad (4)	
USABILIDAD (USA) (72)	USA.1 Facilidad de comprensión (49)	USA.1.1 General (9) USA.1.2 Interactividad (11) USA.1.3 Diseño de la interfaz (24) USA.1.4 Guías didácticas (5)
	USA.2 Capacidad de uso(8) USA.3 Interfaz Gráfica (9) USA.4 Operabilidad (6)	
FIABILIDAD (FIA) (14)	FIA.1 Madurez (6) FIA.2 Recuperación (3) FIA.3 Tolerancia a fallas (5)	
Total de métricas: 168 Total de fórmulas para cuantificar las métricas: 27		

Tabla 1: Resultados de las actividades para adaptar a MOSCA al software educativo.

3.6 Descripción del modelo propuesto.

La propuesta del modelo de evaluación de calidad de software educativo consiste en un conjunto de categorías, características, subcaracterísticas y métricas (ver Figura). La estructura del modelo consta de cuatro niveles que se explican brevemente a continuación:

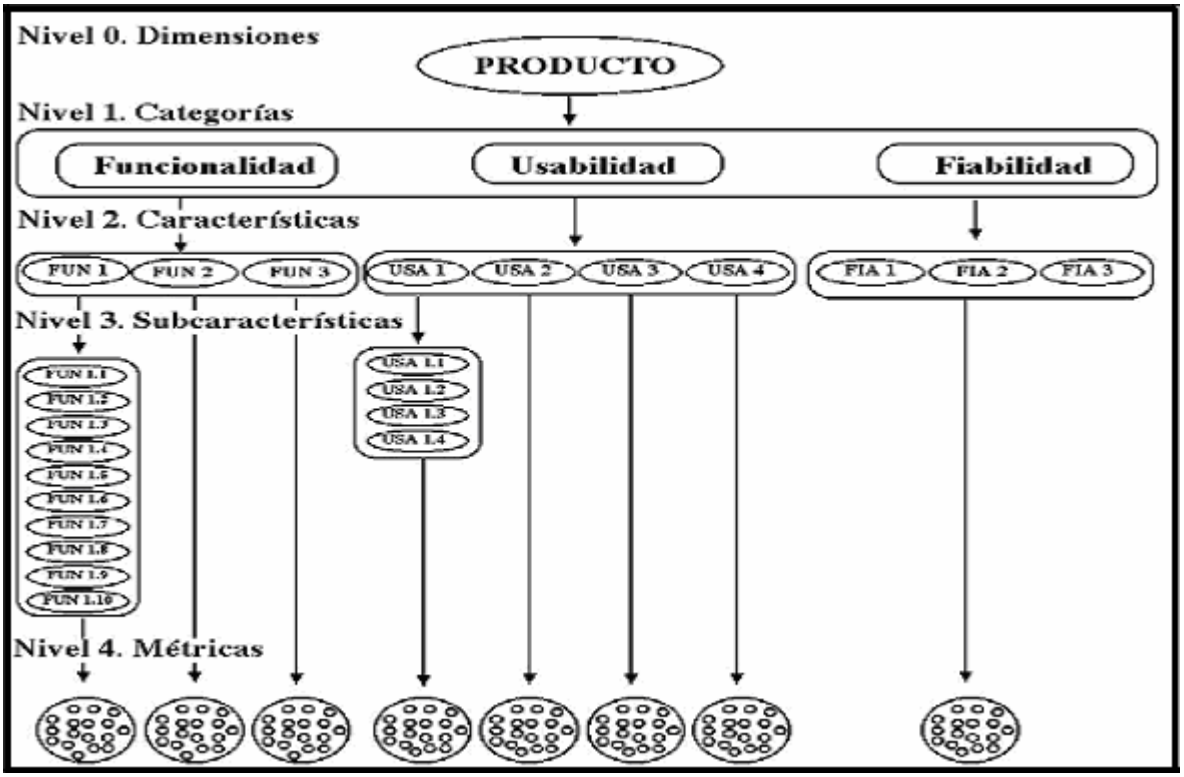


Figura 7: Propuesta del modelo de evaluación de software educativo.

Nivel 0: Dimensiones. Producto.

Nivel 1: Categorías. Se contemplan tres categorías:

Funcionalidad (FUN): Es la capacidad del producto del software para proveer funciones que cumplan con necesidades específicas o implícitas, cuando el software es utilizado bajo ciertas condiciones.

Usabilidad (USA): Esta categoría se refiere a la capacidad del producto de software para ser atractivo, entendido, aprendido y utilizado por el usuario bajo condiciones específicas.

Fiabilidad (FIA): La fiabilidad es la capacidad del producto de software para mantener un nivel especificado de rendimiento cuando es utilizado bajo condiciones especificadas.

Nivel 2: Características. Cada categoría tiene asociado un conjunto de características.

Nivel 3: Subcaracterísticas. Para algunas de las características se asocian un conjunto de subcaracterísticas.

Nivel 4: Métricas. Para cada característica se propone una serie de métricas utilizadas para medir la calidad sistémica.

En resumen, la propuesta del modelo de evaluación de software educativo consta de un total de 3 categorías, 10 características, 14 subcaracterísticas, 168 métricas y 27 fórmulas para cuantifiicar las métricas de cada categoría, característica y subcaracterística. Una vez formulado el modelo, se presenta SIMETSE, una propuesta de métricas para la evaluación de software educativo en la UCI.

3.7 Definición de SIMETSE- SIstema de METricas para evaluar el Software Educativo.

Para la elaboración de SIMETSE, se analiza en primer lugar El Modelo Sistémico de Calidad de Software (MOSCA). En segundo lugar, se selecciona un conjunto del total de las características, categorías de MOSCA, que se ajusten a la evaluación de software educativo de la UCI, formando así la base de la propuesta. Por último se le añaden los parámetros específicos relacionados con la calidad educativa del software, dando lugar a la adición de un nuevo conjunto de medidas que involucran tanto un nuevo parámetro de medición (subcaracterísticas). Con esta ampliación de MOSCA para el software educativo, en cuanto a categorías, características y sub-características se le añaden nuevas métricas para evaluar el software educativo que se elabora en la UCI.

3.7.1 Estimar la calidad del producto de software educativo con enfoque sistémico.

Para calcular el valor de calidad que obtiene cada categoría, característica y subcaracterística se utilizó la siguiente formula general:

$$M = [(\Sigma_{j=1}^{m} (\Sigma_{i=1}^{n} e_{ij} * p)) * 100]/n*m$$

Donde:

M: Valor total de las métricas.

j: Número de columnas.

i: Números de filas.

m: Cantidad de ponderaciones.

n: Cantidad de métricas.

$e_{ij:}$ Elemento de la fila i y columana j.

p: Valor de ponderación.

Para cada Subcaracterística las métricas toman valores entre 1 y 5:

1	2	3	4	5
Nunca	Pocas veces	Algunas veces	Casi siempre	Siempre
Muy mal	Mal	Normal	Bueno	Muy bueno
Nada	Alguno	Normal	Casi	Mucho
No tiene	Básico	Mediano	Alto	Muy alto
Ninguno	Muy pocos	Pocos	Casi todos	Todos
Muy baja	Baja	Media	Alta	Muy alta
No	Poco	Medianamente	Casi todo	Completamente
Inaceptable	Debajo del promedio	promedio	Buena	Excelente
No		Promedio		Si

Figura 8: Normalización de las métricas para la evaluación del producto.

- 1: Valor mínimo. Se le asigna este valor a la métrica cuando el producto no presenta el parámetro que se evalúa o presenta un grado de satisfacción muy bajo.
- 2: Se le asigna este valor a la métrica cuando el producto presenta el parámetro a evaluar pero con un grado de satisfacción bajo.
- 3: Se le asigna este valor a la métrica cuando el producto presenta el parámetro a evaluar con un grado de satisfacción medio.
- 4: Se le asigna este valor a la métrica cuando el producto presenta el parámetro a evaluar con un grado de satisfacción alto.

- 5: Valor máximo. Se le asigna este valor a la métrica cuando el producto cumple con el parámetro establecido y presenta un grado de satisfacción muy alto.

Para cada categoría y características las métricas toman valores entre 0 y 1:

- 0: Cuando no presenta calidad o la categoría no es satisfecha.
- 1: Cuando presenta calidad o la categoría es satisfecha.

Para que el producto obtenga la calidad final debe cumplir con las siguientes actividades:

1. Estimar la calidad de la Funcionalidad del producto: Según MOSCA, se establece que siempre y en todos los casos se debe medir primero la categoría Funcionalidad del producto, donde las características que se proponen para esta categoría deben ser cumplidas para proceder a la próxima actividad. De lo contrario, la evaluación finaliza. Para la categoría funcionalidad se propone que al menos se cumpla la característica "Ajuste a los propósitos", más una de las dos características restantes, es decir, "Precisión" o "Seguridad" (ver ilustración 9).

Categorías del producto	Características mínimas que deben ser satisfechas
Funcionalidad	1. Ajuste a los propósitos 2. Precisión o Seguridad
Usabilidad	3 (de 4)
Fiabilidad	2 (de 3)

Figura 9: Características mínimas que deben ser satisfechas para cada categoría.

2. Estimación de calidad para cada categoría: Para las dos (2) categorías seleccionadas previamente (Usabilidad y Fiabilidad), se debe:

   2.2.1. Aplicar las métricas propuestas para Usabilidad y Fiabilidad.

   2.2.2. Normalizar los resultados de las métricas a una escala del 1 al 5. La normalización de los resultados es llevada a cabo de acuerdo a la figura 9.

   2.2.3. Evaluar la categoría. Una categoría es satisfecha si entra en el rango propuesto. Tratándose de software educativo, donde existen características que son imprescindibles que estén presentes, se proponen las características mínimas satisfechas que debe tener cada categoría del producto para que ésta pueda ser satisfecha.

3. Estimar la calidad del producto partiendo de las categorías evaluadas. En este punto se utiliza el algoritmo de MOSCA, con ciertas modificaciones (ver figura 10). Nótese que para que un software educativo obtenga un nivel de calidad adecuado, deben ser satisfechas las categorías de Funcionalidad y Usabilidad.

Funcionalidad	Usabilidad	Fiabilidad	Nivel de calidad
Satisfecha	No satisfecha	No satisfecha	NULO
Satisfecha	No satisfecha	Satisfecha	NULO
Satisfecha	Satisfecha	No satisfecha	ADECUADO
Satisfecha	Satisfecha	Satisfecha	SATISFACTORIO

Figura 10: Nivel de calidad del producto con respecto a las categorías satisfechas para el producto software educativo.

Una vez terminada la evaluación del producto, y sólo en caso de que éste obtenga al menos el nivel de calidad adecuado:

1. Se procederá a la toma de decisión de la entrega del software educativo a los institutos o empresas.

2. Se procederá a evaluar la calidad del proceso.

3.7.2 Métricas.

A continuación se presentan las métricas relacionadas con las tres categorías: Funcionalidad, Usabilidad y Fiabilidad.

1. Métricas de Funcionalidad.

La funcionalidad es la capacidad del producto de software para proveer funciones que cumplan con necesidades específicas o implícitas, cuando el software es utilizado bajo condiciones específicas. Un software educativo debe ajustarse a una diseño instruccional específico. La funcionalidad toma en cuenta el ajuste a los propósitos, la precisión, y la seguridad del producto de software.

1.1 Ajuste a los propósitos.

La capacidad del producto de software para proveer un conjunto de funciones apropiado según tareas y objetivos específicos del usuario.

1.1.1 General.

№	Métricas. General	1	2	3	4	5
1	Considera que el software ofrece la información apropiada.					
2	Es el software adecuado para el nivel de enseñanza que se le pide.					
3	Es de fácil comprensión y entendimiento.					
T						

Tabla 2: Tabla de métricas para la subcaracterística: General.

Métrica.

$$M= [(\Sigma_{j=1}^{5} (\Sigma_{i=1}^{3} eij * p))*100]/15$$

Rango: 60 ≤ M≤100

1.1.2 Objetivos de aprendizaje.

№	Métricas. Objetivos de aprendizaje	1	2	3	4	5
1	Ayudan los objetivos a entender el contenido.					
2	Abarcan los objetivos puntos importantes del tema en cuestion.					
3	Son los objetivos adecuados según a quien van dirigidos.					
4	Estan bien definidos los objetivos para su mayor entendimiento.					
5	Facilitan los objetivos la comprensión de los conceptos.					
T						

Tabla 3: Tabla de métricas para la subcaracterística: Objetivos de aprendizaje.

Métrica.

$$M = [(\sum_{j=1}^{5} (\sum_{i=1}^{5} e_{ij} * p))*100]/25$$

Rango: $60 \leq M \leq 100$

1.1.3 Contenidos de aprendizaje.

№	Métricas. Contenidos de aprendizaje	1	2	3	4	5
1	Concuerda el contenido con los objetivos trazados.					
2	Esta el contenido fácil de entender.					
3	Ayuda el contenido al aprendizaje.					
4	Considera que el software educativo le da la información requerida.					
5	El contenido va en contra de los principios morales.					
6	Presenta el contenido rasgos de pobreza.					
7	El contenido esta acorde con la edad a la que va dirigido.					
8	Presenta el contenido un nivel de coherencia adecuado.					
9	Presenta el contenido errores ortográficos o de gramática.					
10	Proporciona beneficios.					
11	Son los contenidos adecuados para la comprensión del tema.					
12	La información que presenta es actualizada y rigurosa.					
T						

Tabla 4: Tabla de métricas para la subcaracterística: Contenidos de aprendizaje.

Métrica.

$$M = [(\Sigma_{j=1}^{5} (\Sigma_{i=1}^{12} e_{ij} * p))*100]/60$$

Rango: $55 \leq M \leq 100$

1.1.4 Actividades de aprendizaje.

№	Métricas. Actividades de aprendizaje.	1	2	3	4	5
1	Están en correspondencia con los objetivos trazados.					
2	Están en correspondencia con el contenido.					
3	Permiten comprobar si el contenido fue dominado.					
4	Considera adecuada la selección de las actividades.					
5	Presenta diferentes actividades.					
6	Son suficientes las actividades presentadas en el software para evaluar el contenido.					
7	Permiten hacer verificaciones en caso de contestar incorrectamente.					
8	Se va incrementando el nivel de las actividades según lo que avanzas.					
9	Le resulta útil el uso de las actividades.					
10	Permite el seguimiento por el usuario de su trabajo.					
11	Las preguntas evaluativas son aleatorias y permiten comprobar el dominio de los objetivos.					
T						

Tabla 5: Tabla de métricas para la subcaracterística: Actividades de aprendizaje.

Métrica.

$$M = [(\Sigma_{j=1}^{5} (\Sigma_{i=1}^{11} e_{ij} * p))*100]/55$$

$$j=1 \quad i=1$$

Rango: $55 \leq M \leq 100$

1.1.5 Ejemplos.

№	Métricas. Ejemplos	1	2	3	4	5
1	Se muestran ejemplos según lo requiere el contenido.					
2	Son los ejemplos fáciles de comprender.					
3	Están acorde con el contenido.					
T						

Tabla 6: Tabla de métricas para la subcaracterística: Ejemplos.

Métrica.

$$M= [(\Sigma_{j=1}^{5} (\Sigma_{i=1}^{3} e_{ij} * p))*100]/15$$

Rango: $60 \leq M \leq 100$

1.1.6 Motivación.

№	Métricas. Motivación.	1	2	3	4	5
1	El contenido motiva estudiante.					
2	Presenta aspectos que lo distingue.					
3	El estudiante le concede mayor interes que a otro.					
4	Lo utilizan con frecuencia aunque ya rebasaron los conocimientos.					
5	Disfruta utilizarlo.					
6	Le brinda al estudiante opciones por las cuales el se podria sentir motivado.					

7	Estimula al estudiante.
8	Despierta el interés.
9	Se establece competencias entre los estudiantes.
10	Los grupos, o el aula en general se siente motivada a utilizarlo.
T	

Tabla 7: Tabla de métricas para la subcaracterística: Motivación.

Métrica.

$$M = [(\Sigma_{j=1}^{5} (\Sigma_{i=1}^{10} e_{ij} * p))*100]/50$$

Rango: $65 \leq M \leq 100$

1.1.7 Retro-alimentación.

№	Métricas. Retro-alimentación.	1	2	3	4	5
1	Es adecuada.					
2	Es amena.					
3	Ayuda al estudiante cuando es necesario.					
4	El software corrige con eficacia.					
5	Te presenta la oportunidad de corregir las respuestas erróneas.					
6	Te ofrece respuestas ante cualquier fallo o error.					
7	Te ayuda a elevar tu optimismo.					
T						

Tabla 8: Tabla de métricas para la subcaracterística: Motivación.

Métrica.

$$5 \quad 7$$

$$M = [(\Sigma_{j=1} (\Sigma_{i=1} e_{ij} * p))*100]/35$$

Rango: $60 \leq M \leq 100$

1.1.8 Ayudas

№	Métricas. Ayudas.	1	2	3	4	5
1	Contiene un glosario de ayuda.					
2	Brinda datos que pueden mejorar los conociminetos.					
3	Ayuda a entender como se utiliza.					
T						

Tabla 9: Tabla de métricas para la subcaracterística: Ayudas.

Métrica.

$$M = [(\Sigma_{j=1}^{5} (\Sigma_{i=1}^{3} e_{ij} * p))*100]/15$$

Rango: $60 \leq M \leq 100$

1.1.9 Evaluación y registro de calificaciones

№	Métricas. Evaluación y registro de calificaciones.	1	2	3	4	5
1	Permite la posibilidad de registrar las calificaciones.					
2	Muestra cada calificación al terminar una actividad.					
3	Muestra como vas venciendo el contenido.					
4	Orienta al usuario.					
5	Le da buena información acerca del desempeño del usuario.					
6	Permite tomar varias decisiones según sea posible.					

7	Los criterios de evaluación son adecuados según el contenido.
8	Interactúa con el usuario mostrándole información útil.
9	Es adecuada las calificaciones.
10	Posibilita registrar las calificaciones.
T	

Tabla 10: Tabla de métricas para la subcaracterística: Evaluación y registro de calificaciones.

Métrica.

$$M= [(\Sigma_{j=1}^{5} (\Sigma_{i=1}^{10} e_{ij} * p))*100]/50$$

Rango: $65 \leq M \leq 100$

1.1.10 Metodología de la enseñanza.

№	Métricas. Metodología de la enseñanza.	1	2	3	4	5
1	Utiliza los principios metodológicos.					
2	Contribuye a un aprendizaje activo y significativo.					
3	Es la metodología adecuada para el aprendizaje del tema.					
4	La pedagogía es adecuada.					
5	Ayuda al usuario a participar activamente.					
6	Ayuda al usuario a desarrollar su pensamiento lógico.					
7	Utiliza diálogos, videos u otras alternativas para comunicarse con el usuario.					
8	Relación entre contenido, características del estudiante y estrategia metodológica.					

9	Fomenta la creatividad del usuario.	
T		

Tabla 11: Tabla de métricas para la subcaracterística: Metodología para la enseñanza.

Métrica.

$$M = [(\Sigma_{j=1}^{5} (\Sigma_{i=1}^{9} e_{ij} * p))*100]/45$$

Rango: $60 \leq M \leq 100$

Característica: Ajuste a los propósitos

№	Métricas. Ajuste de los propósitos.	0	1
1	General.		
2	Objetivos de aprendizaje.		
3	Contenidos de aprendizaje.		
4	Actividades de aprendizaje.		
5	Ejemplos.		
6	Motivación.		
7	Retro-alimentación.		
8	Ayudas.		
9	Evaluación y registro de calificaciones.		
10	Metodología de la enseñanza.		
T			

Tabla 12: Para calcular la característica: Ajuste a los propósitos.

Métrica.

$$M = [(\Sigma_{j=1}^{1} (\Sigma_{i=1}^{10} e_{ij} * p))*100]/45$$

$$M= [(\Sigma_{j=1} (\Sigma_{i=1} eij * p))*100]/10$$

Rango: 65≤M≤100

1.2 Precisión.

La capacidad del producto de software para proveer los resultados correctos. Esto incluye el grado de precisión de los valores calculados.

№	Métricas. Precisión	1	2	3	4	5
1	Presenta errores a la hora de mostrar los resultados.					
2	Presenta errores en el contenido.					
3	Emite resultados o mensajes no esperados que no presentan relación alguna.					
4	Presenta resultados incorrectos.					
5	Muestra las calificaciones correctamente.					
T						

Tabla 13: Tabla de métricas para la característica: Presición.

Métrica.

$$M= [(\Sigma_{j=1}^{5} (\Sigma_{i=1}^{5} eij * p))*100]/25$$

Rango: 60≤M≤100

1.3 Seguridad.

La capacidad del producto de software para proteger información y datos de manera que las personas que no estén autorizadas no puedan leerlos o modificarlos y a las personas o sistemas autorizados se les permita el acceso a los mismos.

№	Métricas. Seguridad.	1	2	3	4	5
1	No permite el acceso a la información de otro estudiante.					
2	No permite modificar las calificaciones almacenadas.					
3	El software educativo registra y detecta el acceso del usuario al sistema.					
4	No hay posibilidad de modificar el contenido o alterarlo.					
T						

Tabla 14: Tabla de métricas para la característica: Seguridad.

Métrica.

$$M= [(\Sigma_{j=1}^{5} (\Sigma_{i=1}^{4} e_{ij} * p)) *100]/20$$

Rango: $60 \leq M \leq 100$

Categoría: Funcionalidad.

№	Métricas. Funcionalidad.	0	1
1	Ajuste a los propósitos.		
2	Presición.		
3	Seguridad.		
T			

Tabla 15: Para la categoría: Funcionalidad.

Métrica.

$$M= [(\Sigma_{j=1}^{1} (\Sigma_{i=1}^{3} e_{ij} * p)) *100]/3$$

j=1 i=1

Rango: 65≤M≤100

2. Métricas de Usabilidad.

La capacidad del producto de software para ser atractivo, entendido, aprendido, y utilizado por el usuario bajo condiciones específicas.

2.1 Facilidad de comprensión del software para el proceso de aprendizaje.

La capacidad que tiene el producto de software para facilitar al usuario el entender el software y la forma en que puede ser utilizado para efectuar diferentes tareas bajo condiciones específicas. Estas métricas deben ser capaces de evaluar el comportamiento de los usuarios sin previo conocimiento de la operación del software y medir la dificultad al entender las funciones, operaciones y conceptos del software. Pueden ser consideradas entidades como documentación (en todos los formatos disponibles, en línea o impreso), interfaces de software y vocabulario.

2.1.1 General

№	Métricas. General.	1	2	3	4	5
1	El software educativo es de fácil manejo.					
2	Es adecuado el uso del tipo de letras, botones, ventanas.					
3	El usuario adquiere los conocimientos necesarios para vencer el contenido.					
4	Le crea destreza al usuario.					
5	Le ayuda en sus conocimientos.					
6	Ayuda al usuario a razonar por si mismo.					
7	Permite al usuario corregir sus respuestas.					
8	Presenta herramientas que facilitan el aprendizaje.					
9	Es adecuado el diseño general de la pantalla.					

T

Tabla 16: Tabla de métricas para la subcaracterística: General.

Métrica.

$$M = [(\Sigma_{j=1}^{5} (\Sigma_{i=1}^{9} e_{ij} * p)) * 100]/45$$

Rango: $60 \leq M \leq 100$

2.1.2 Interactividad.

La interactividad entre dos sistemas implica que cada uno de ellos es capaz de actuar y reaccionar. El paquete de software educativo será interactivo en la medida que sea variado y adaptable de acuerdo a las diferentes respuestas de sus usuarios y si les permite afectar o modificar la manera en que el software procede.

Métricas Con respecto al usuario y al software.

№	Métricas. Interactividad.	1	2	3	4	5
1	Tiene un mapa que facilite la navegación.					
2	Puede entrar y salir cada vez que desee.					
3	Tiene que seguir un orden o puede saltar a la parte que le interese.					
4	Permite ubicar al usuario en donde esta y cuanto le falta.					
5	Puede parar el programa y salir de él en cualquier momento.					
6	Presenta un menú para escoger lo que desea.					
7	Hay que responder las preguntas según aparezcan o busca las preguntas que desee el usuario.					

8	Facilita que salte la secuencia lógica para donde desee ir el usuario.
9	Ayuda al usuario a contestar las preguntas.
10	Presenta una interfaz amigable.
11	Es interactivo.
T	

Tabla 17: Tabla de métricas para la subcaracterística: Interactividad.

Métrica.

$$M= [(\Sigma_{j=1}^{5} (\Sigma_{i=1}^{11} eij * p)) *100]/55$$

Rango: $55 \leq M \leq 100$

2.1.3 Diseño de la interfaz.

El diseño de la interfaz está relacionado con el uso del espacio y la manera en que la información está dispuesta en la pantalla. Su propósito es hacer que las interacciones entre el usuario y el software sean más fáciles mediante la presentación de la información de una manera adecuada. La disposición de la pantalla es extremadamente importante en términos de motivar al usuario y estructurar la información. Los siguientes aspectos están incluidos: organización del texto en la interfaz, los gráficos, el uso del color y el uso del sonido.

№	Métricas. Diseño de la interfaz.	1	2	3	4	5
1	El texto esta en correspondencia con el contenido.					
2	El texto es claro y de fácil comprensión.					
3	Cuando contiene palabras rebuscadas le ayuda al usuario a entenderlas.					
4	El texto permite leer de manera lógica y sistemática.					

5	Es coherente.
6	Los caracteres son claros y atractivos.
7	El color de la letra es adecuado para la lectura.
8	El tamaño es el adecuado.
9	Presenta palabras calientes con sus significados.
10	La estructura de las oraciones y párrafos están de forma correcta.
11	El fondo de la pantalla es el correcto.
12	Los colores utilizados van acorde con el tema.
13	Presenta gráficos que ayudan a entender un tema en particular.
14	Las ilustraciones e imágenes ayudan al usuario a entender lo que esta leyendo.
15	Las ilustraciones e imágenes están acorde con el tema.
16	Los gráficos ayudan a entender el contenido expuesto.
17	Presenta colores claros que ayudan al usuario a no desviarse del contenido.
18	La combinación de colores es variada, agradable.
19	El contraste entre: los colores de fondo, la letra, gráficos e ilustraciones es adecuado.
20	Hay suficiente contraste entre los colores del texto o ilustraciones y el color del fondo.
21	Los videos y animaciones posee una calidad adecuada.
22	Los videos tienen relación con el contenido.

23	La calidad del sonido es adecuada.
24	Se puede controlar el volumen de los videos.
T	

Tabla 18: Tabla de métricas para la subcaracterística: Diseño de la interfaz.

Métrica.

$$M = [(\Sigma_{j=1}^{5} (\Sigma_{i=1}^{24} e_{ij} * p)) * 100]/120$$

Rango: $60 \leq M \leq 100$

2.1.4 Guías didácticas (Documentación y material de apoyo educativo)

Aunque los programas sean fáciles de utilizar y auto explicativos, conviene que tengan una información que informe detalladamente de sus posibilidades didácticas. Esta documentación (on-line o en papel) debe tener una presentación agradable, con textos bien legibles y adecuados a sus destinatarios, y resultar útil, clara, suficiente y sencilla.

№	Métricas. Guías didácticas.	1	2	3	4	5
1	Explica los objetivos y el contenido de manera adecuada.					
2	Presenta una guía que ayude al usuario.					
3	Incluye bibliografía.					
4	Proporciona sugerencias de cómo navegar.					
5	Sugiere el uso de otros materiales de enseñanza.					
T						

Tabla 19: Tabla de métricas para la subcaracterística: Guías didácticas.

Métrica.

$$M = [(\Sigma_{j=1}^{5} (\Sigma_{i=1}^{5} e_{ij} * p)) * 100]/25$$

j=1 i=1

Rango: 60≤M≤100

Característica: Facilidad de comprensión.

No	Métricas. Facilidad de comprensión.	0	1
1	General.		
2	Interactividad.		
3	Diseño de la interfaz.		
4	Guías didácticas.		
T			

Tabla 20: Para la característica: Facilidad de comprensión.

Métrica.

$$M= [(\Sigma_{j=1}^{4} (\Sigma_{i=1} eij * p)) *100]/4$$

Rango: 75≤M≤100

2.2 Capacidad para el aprendizaje del uso del software educativo.

La capacidad del producto de software para habilitar al usuario el aprendizaje de la aplicación. Aunque los programas sean fáciles de utilizar y auto explicativos, conviene que tengan una información que informe detalladamente de sus características, instalación y forma de uso. Esta documentación (on-line o impresa) debe tener una presentación agradable, con textos bien legibles y adecuados a sus destinatarios, y resultar útil, clara, suficiente y sencilla.

№	Métricas. Capacidad de uso.	1	2	3	4	5
1	El diseño es adecuado para el usuario que debe utilizar el software.					

2	Presenta una información clara y sencilla de cómo instalarlo.
3	Es de fácil manejo.
4	Informa al usuario de las características particulares que presenta.
5	Orienta al usuario a la hora de lidiar con el.
6	Presenta guías de uso.
7	Las guías de uso están redactadas de una manera comprensible.
8	Te da la posibilidad de consultar alguna información en caso de algún problema con el software.

T

Tabla 21: Tabla de métricas para la característica: Capacidad de uso.

Métrica.

$$M= [(\Sigma_{j=1}^{5} (\Sigma_{i=1}^{8} eij * p)) *100]/40$$

Rango: $60 \leq M \leq 100$

2.3. Atractivo de la interzas gráfica.

La capacidad del producto de software para ser atractivo al usuario. Está asociada a los atributos de la interfaz (la pantalla) que hacen que sea más atractivo al usuario y más fácil de usar.

№	Métricas. Interfaz gráfica.	1	2	3	4	5
1	El diseño de la interfaz es adecuado para el usuario al que va dirigido.					
2	El diseño de la interfaz es atractivo.					
3	Considera adecuado el diseño general de la pantalla.					

4	La interfaz es amigable.	
5	La interfaz no esta sobrecargada.	
6	La interfaz es eficiente para el intercambio de información entre el usuario y el programa	
7	El diseño de la interfaz evita la perdida de tiempo.	
8	Las opciones se localizan rápidamente	
9	Es rápida la navegación	
T		

Tabla 22: Tabla de métricas para la característica: Interfaz gráfica.

Métrica.

$$M = [(\Sigma_{j=1}^{5} (\Sigma_{i=1}^{9} eij * p)) * 100] / 45$$

Rango: $60 \leq M \leq 100$

2.4. Operabilidad.

La capacidad del producto de software para habilitar al usuario a operarlo y controlarlo.

№	Métricas. Operabilidad.	1	2	3	4	5
1	Velocidad de acceso.					
2	Provee funciones adicionales para ayudar al usuario.					
3	Provee al usuario de instrucciones indicándole que debe hacer.					
4	El menú presenta una secuencia ordenada.					
5	Existe una correcta operación del software en el ambiente estipulado para la computadora en el manual de uso (tipo de sistema operativo, Windows 95, 98, 2000, Macintosh					

	OS, etc.).		
6	Existe información extra que pueda ayudar al usuario como operar el software.		
T			

Tabla 23: Tabla de métricas para la característica: Operabilidad.

Métrica.

$$M= [(\Sigma_{j=1}^{5} (\Sigma_{i=1}^{6} eij * p)) *100] /30$$

Rango: 65≤M≤100

Categoría: Usabilidad.

№	Métricas. Usabilidad.	0	1
1	Facilidad de comprensión.		
2	Capacidad de uso.		
3	Interfaz Gráfica.		
4	Operabilidad.		
T			

Tabla 24: Para la categoría: Usabilidad.

Métrica.

$$M= [(\Sigma_{j=1}^{1} (\Sigma_{i=1}^{4} eij * p)) *100] /4$$

Rango: 75≤M≤100

3. Métricas de Fiabilidad.

Es la capacidad del producto de software para mantener un nivel especificado de rendimiento cuando es utilizado bajo condiciones especificadas.

3.1 Madurez

La capacidad del producto de software para evitar fallas como resultado de errores en el software. Se pueden encontrar errores tales como: no reconoce el usuario, el usuario no encuentra su registro, el programa no responde a un comando, el programa produce un error en el sistema operativo de la computadora.

№	Métricas. Madurez.	1	2	3	4	5
1	El software educativo no presenta fallas.					
2	En caso de que ocurra algún error o falla es solucionado inmediatamente.					
3	Guarda registro de las fallas después que ocurren.					
4	Hay información sobre los requerimientos de hardware.					
5	Los datos proporcionados de hardware y software son compatibles con los equipos del laboratorio de computación.					
6	Los periféricos (no incluidos en el paquete) que son difíciles de adquirir o son demasiado caros no son indispensables.					
T						

Tabla 25: Tabla de métricas para la característica: Madurez.

Métrica.

$$M = [(\Sigma_{j=1}^{5} (\Sigma_{i=1}^{6} e_{ij} * p)) * 100] / 30$$

Rango: $65 \leq M \leq 100$

3.2 Tolerancia a fallas.

La capacidad del producto de software para mantener un nivel de rendimiento especificado en caso de errores en el software o de infracciones sobre sus interfaces.

№	Métricas. Tolerancia a fallas.	1	2	3	4	5
1	Ocurren fallas frecuentemente.					
2	Se le da una explicación en caso de fallas, para la solución.					
3	Soluciona las fallas que ocurren.					
4	Le da la opción al usuario de solucionar las fallas en caso de que el mismo no pueda.					
5	Estas opciones de solución resuelven realmente el problema.					
T						

Tabla 26: Tabla de métricas para la característica: Tolerancia a fallas.

Métrica.

$$M = [(\sum_{j=1}^{5} (\sum_{i=1}^{5} e_{ij} * p)) * 100] / 25$$

Rango: $60 \leq M \leq 100$

3.3 Recuperación.

La capacidad del producto de software para restablecer un nivel especificado de rendimiento y recuperar los datos afectados en el caso de una falla. Después de una falla, el producto de software algunas veces se encuentra no operativo por un período de tiempo, el lapso de este tiempo es medido como su recuperación. Una definición asociada a la recuperación es la disponibilidad, que consiste en la capacidad de un producto de software para estar en estado operativo en el momento de ejecutar una función en un momento determinado, bajo condiciones específicas. Cuando el software se cierra inesperadamente, se dice que es una falla. Con esta categoría se determina si es necesario reiniciar la computadora, o solo volver a iniciar el programa de una manera rápida o que toma mucho tiempo en animaciones o instrucciones que hay que seguir.

№	Métricas. Recuperación.	1	2	3	4	5
1	Es bueno el funcionamiento del sistema después de una falla					
2	Se recupera el sistema después de una falla.					
3	La velocidad con que se recupera es rápida.					
T						

Tabla 27: Tabla de métricas para la característica: Recuperación.

Métrica.

$$M = [(\Sigma_{j=1}^{5} (\Sigma_{i=1}^{3} eij * p)) * 100] / 15$$

Rango: $60 \leq M \leq 100$

Categoría: Fiabilidad.

№	Métricas. Fiabilidad.	0	1
1	Madurez.		
2	Tolerancia a fallas.		
3	Recuperación.		
T			

Tabla 28: Para la categoría: Fiabilidad.

Métrica.

$$M = [(\Sigma_{j=1}^{1} (\Sigma_{i=1}^{3} eij * p)) * 100] / 3$$

Rango: $65 \leq M \leq 100$

3.8 Evaluación mediante Juicio de Expertos.

Como parte del ciclo de desarrollo del software, es importante verificar si los objetivos previstos se han cumplido en la práctica o si los ajustes que se introdujeron en el desarrollo efectivamente mejoran la calidad del diseño y para hacer esto es conveniente recurrir a especialistas de la materia que sean preferiblemente ajenos al centro de elaboración del software en aras de ganar objetividad

Cada uno de los expertos consultados, velará por la eficacia y eficiencia del software educativo, realizando una valoración de los aspectos funcionales, técnicos, estéticos y pedagógicos. Desde el punto de vista metodológico opinarán sobre si el tratamiento es consistente con la didáctica que es deseable para promover el logro de los objetivos.

Los especialistas darán su opinión sobre el entorno de trabajo utilizado, el sistema de motivación y de refuerzo, la forma como se llega al conocimiento, el sistema de evaluación y de reorientación. También considerará el cumplimiento de las funciones de apoyo para cada tipo de usuario, determinará si los objetivos, contenidos y tratamiento responden a la necesidad que pretende satisfacer el software educativo, si las funciones de apoyo relacionadas con el contenido para cada tipo de usuario se cumple a cabalidad; y se pronunciarán sobre la actualidad, pertinencia, exactitud y completitud del contenido y de los ejemplos y ejercicios, dentro del entorno de trabajo en el que se presente.

Según la bibliografía consultada la autora considera y está de acuerdo en que la revisión por expertos no implica, necesariamente, que el software educativo va a funcionar apropiadamente y producir los resultados esperados al ser usado por los distintos destinatarios. Tan solo se tiene una alta probabilidad de que así sea, pero habrá que probar el software educativo por usuarios reales.

3.9 Conclusiones.

Se hizo un estudio del modelo sistémico de calidad (MOSCA), tomándose este para adaptarlo al área de software educativa, se definió un conjunto de métricas para evaluar la calidad del software en la UCI, estas métricas favorecerán la calidad de los productos, y brindarán una medida de cómo esta progresando el desarrollo del software educativo. Servirán como base para medir el software educativo que se produce en la Universidad.

CONCLUSIONES.

La presencia de las Tecnologías de la Información y las Comunicaciones (TIC) en el aula, es cada día mayor; sin embargo, es un reto garantizar que se haga un uso adecuado de las mismas. Parte de este reto es contar con software de calidad. En particular, con el software que esté soportando el proceso de enseñanza-aprendizaje. Con el objetivo de este trabajo de diploma cumplido se arriban a las siguientes conclusiones:

- La calidad es un elemento multidimensional y contextual, lo cual hace al software altamente cualitativo.
- La buena calidad de software se puede tener cuando los programas tienen menos impurezas.
- No se aplican métricas en la UCI para evaluar la calidad de los productos y hay un desconocimiento casi general de estas.
- El Modelo Sistémico de Calidad (MOSCA), se adapta al software educativo que se produce en la UCI.
- Utilizando SIMETSE se puede medir la calidad del software educativo, por lo que ayudará a aumentar la calidad de dichos software en la UCI, permitiendo a la universidad contar con instrumento orientador, que cuantifique (mide) la calidad del software educativo.

RECOMENDACIONES

Este trabajo de diploma propuso un sistema de métricas para evaluar el software educativo (SIMETSE) se recomienda:

- Publicar los resultados de este trabajo de diploma para poner a disposición de la UCI el sistema de métricas propuesto.
- Aplicar SIMETSE en algún proyecto productivo.
- Proponer, tras comprobar un resultado exitoso, la utilización de SIMETSE en el desarrollo de software educativo de la UCI.

SIMETSE solo toma algunas de las métricas que se definen en MOSCA por lo que en próximos trabajos se podria

- Aumentar el nivel cuatro del sistema de métricas.
- Continuar el estudio del tema con el objetivo de actualizar el sistema propuesto.

BIBLIOGRAFÍA

1. Álvarez, J.L.M., Aplicación de un Sistema Experto para el desarrollo de Sistema Evaluador del modelo Capability Maturity Model (CMM) niveles dos y tres 2004, Universidad de las Américas: México.

2. Rodríguez, L.D.A.C., L.R.S. Nadal, and D.A.M.G. Pérez AUTOMATIZACIÓN DE LA GESTIÓN DE LA CALIDAD DE UNA ORGANIZACIÓN DE SOFTWARE PARTIENDO DE LA MEDICION DEL TAMAÑO. Volume.

3. Perez, M., et al. Calidad Sistémica del Software Educativo. Volume.

4. Informática, D.d.C.d.C.y.A., Control de Calidad en los Sistemas. 2000: p. 52.

5. Gros, B., Del software educativo a educar con software. Quaderns Digital, 2000.

6. Callaos, N. and B. Callaos. Designing with Systemic Total Quality. in International Conference on Information Systems. 1993: International Institute of Informatics and Systemics.

7. I.Jacobson, G.Booch, and J. Rumbaugh, El Proceso Unificado de Desarrollo de Software. 2000, Madrid. 464.

8. Galvis, A., Ingeniería de software educativo. 2da. Reimpresión. ed. 2000, Universidad de Los Andes. Colombia

9. Cataldi, Z., et al. INGENIERIA DE SOFTWARE EDUCATIVO. Volume,

10. Pressman., R., Ingeniería de Software: Un enfoque Práctico. Quinta ed. 2002. 614.

11. Victoria-Gasteiz., E.U.d.I. (2006-2006) Ingeniería del Software de Gestión. Tema 1. Volume,

12. Díaz-Antón, G., et al. Instrumento de evaluación de software educativo bajo un enfoque sistémico. Volume.

13. Mendoza, G.M. (2006) ISO 9126-3: Métricas Internas de la Calidad del Producto de Software. Volume.

14. Melian., J.M.M. and J.F.H. Ballesteros., La calidad del software y su medida. .

15. Marquès, P. (1998) La evaluación de programas didácticos. Volume, 53-58

16. Doria, H.G., Las Métricas de Software y su Uso en la Región. , in Departamento de Ingeniería en Sistemas Computacionales, Escuela de Ingeniería,. 2001, Universidad de las Américas-Puebla.

17. Sans, M.C. (1998) Las normas ISO.Volume.

18. Santos, J.M. Mediciones. Enfoques de las métricas. Volume,

19. Rodríguez., F.S. (1999) Medida del Tamaño Funcional de Aplicaciones Software. Volume, 65

20. Estrada, M.L.A.F. Medir el proceso de control de configuración, ¿una utopía para la Industria Nacional de Software? Volume, 18

21. Cataldi, Z., Metodología de diseño, desarrollo y evaluación de software educativo. 2000, Facultad de Informática. UNLP. p. 75.

22. Estévez, I.P., MÉTRICAS PARA EL CONTROL DE PROYECTOS DE SOFTWARE. 2002, INSTITUTO SUPERIOR POLITÉCNICO "JOSÉ ANTONIO ECHEVERRÍA".FACULTAD DE INGENIERÍA INDUSTRIAL. CENTRO DE ESTUDIOS DE INGENIERÍA DE SISTEMAS: Ciudad de la Habana.

23. Giraldo, O.P. Métricas, Estimación y Planificación en Proyectos de Software. Volume,

24. Grimán, A., et al. MODELO DE CALIDAD DE SOFTWARE EDUCATIVO: APLICACIÓN DEL ESTÁNDAR ISO/IEC 9126. Volume,

25. L.Mendoza, M. Pérez, and T.Rojas (2001) Modelo sistémico para estimar la calidad de los sistemas de software. (MOSCA). Volume, 435

26. Organización Internacional para la Estandarización.

27. Hernandez, M.A.M. Personal Software Process/Team Software Process (PSP/TSP). Volume,

28. Cedillo, K. Proceso de Software Personal. PSP. Volume, 39

29. Feijoo, M.G.D.d., PROPUESTA DE UNA METODOLOGÍA DE DESARROLLO Y EVALUACIÓN DE SOFTWARE EDUCATIVO BAJO UN ENFOQUE DE CALIDAD SISTÉMICA. 2002, Universidad Simón Bolívar. p. 153.

30. Bayona, S., et al. (2007) TEAM SOFTWARE PROCESS (TSP): MEJORAS EN LA ESTIMACIÓN, CALIDAD Y PRODUCTIVIDAD DE LOS EQUIPOS EN LA GESTIÓN DEL SOFTWARE. Volume.

31. Dapena, M.D.D., S.Á. Cárdenas, and A.R. Suárez, UNA PROPUESTA DE INTRODUCCIÓN DE LAS REVISIONES EN EL PROCESO DE DESARROLLO DE SOFTWARE. REVISTA INVESTIGACIÓN OPERACIONAL 2005. 26, No. 2.

32. Alonso, L.M.G. UNIDAD II Métricas y Procesos PSP "Personal Software Process". . Volume, 127

REFERENCIAS BIBLIOGRÁFICAS

A.Galvis (2000). Ingeniería de software educativo. Colombia, 2da. reimpresión.

Álvarez, J. L. M. (2004). Aplicación de un Sistema Experto para el desarrollo de Sistema Evaluador del modelo Capability Maturity Model (CMM) niveles dos y tres. México, Universidad de las Américas.

BORK, A. (1986). El ordenador en la enseñanza. Análisis y perspectivas de futuro. Barcelona.

Callaos, N. and B. Callaos (1993). Designing with Systemic Total Quality. International Conference on Information Systems, International Institute of Informatics and Systemics.

Cataldi, Z. (2000). Metodología de diseño, desarrollo y evaluación de software educativo, Facultad de Informática. UNLP.

CUE., R. (2007). Ciudad de la Habana**:** Entrevista personal.

Estévez, I. P. (2002). MÉTRICAS PARA EL CONTROL DE PROYECTOMÉTRICAS PARA EL CONTROL DE PROYECTOS DE SOFTWARE. Ciudad de la Habana, INSTITUTO SUPERIOR POLITÉCNICO "JOSÉ ANTONIO ECHEVERRÍA.

Febles, A. (2000). Case Corporativo para el proceso de control de cambios. Ciudad de la Habana.

FENTON, E. N. (1991). Software Metrics A rigorous approach.

Gros, B. (2000). "Del software educativo a educar con software." Quaderns Digital.

Hayes, W. (1997). The Personal Software Process: An empirical study of the impact of PSP on individual engineers", Carnegie Mellon

Hernández., O. A. (2007). Ciudad de la Habana**:** Entrevista personal.

Humphrey., W. S. (2000.). Introduction to the Team Software Process.

IEEE (1993) "Software Engineering Standards Std. 6 10.12-1990." **Volume**, 47-48 DOI:

ISO/IEC (1991) "ISO/IEC 9126 JTC 1/SC 7. Information technology - Software product evaluation -Quality characteristics and guidelines for their use, JTC 1 Organization." **Volume**, DOI:

K.ISHIKAWA (1998). "¿Qué es el Control Total de la Calidad?: Modalidad Japonesa ".

Kan, S. H. (2000) "Metrics and Models in Software Quality Engineering." **Volume**, DOI:

L.Briand, C. B., J. Daly, C.Differding (1996) "An experimental comparison of the maintainability of object-oriented and structured design documents." **Volume**, DOI:

L.Mendoza, M. Pérez, et al. (2001) "Modelo sistémico para estimar la calidad de los sistemas de software". (MOSCA)." **Volume**, 435 DOI:

Lorente, A. E. (2007). Ciudad de la Habana**:** Entrevista personal.

McGarry, J., E. Bailey, et al. (1998). Practical Software Measurement: A Foundation for Objective Project Management

P.Marquès (1998) "La evaluación de programas didácticos". Comunicación y Pedagogía." **Volume**, DOI:

PRESSMAN, R. (1993). Ingeniería de Software. Un enfoque práctico.

Pressman, R. (2002). Ingeniería de Software: Un enfoque Práctico.

Pressman, R. S. (1998). Ingeniería del software. Un enfoque práctico, 4ª Edición.

R.G.Grady (1992.) "Practical Software Metrics for Project Management and Process Improvement." **Volume**, DOI:

Rodríguez, D. A. C., L. R. S. Nadal, et al. "AUTOMATIZACIÓN DE LA GESTIÓN DE LA CALIDAD DE UNA ORGANIZACIÓN DE SOFTWARE PARTIENDO DE LA MEDICION DEL TAMAÑO." **Volume**, DOI:

S.KAN (1995) "Metrics and Models in Software Quality Engineering." **Volume**, DOI:

Victoria-Gasteiz., E. U. d. I. (2006-2007
) "Ingeniería del Software de Gestión. Tema 1
" **Volume**, DOI:

Westfall, L. L. (1995) "Software metrics that meet your information needs." **Volume**, DOI:
Wiegers, K. E. (1999) "A Software Metrics Primer." **Volume**, DOI:
Yanko Hernández, I. B. (2001). Case para la planificación y control de configuración de software. Ciudad de la Habana, Instituto Superior Politécnico “José Antonio Echeverría.
Zuse, H. (1995) "History of Software Measurement." **Volume**, DOI:

ANEXOS ANEXO1 NIVELES DEL PSP

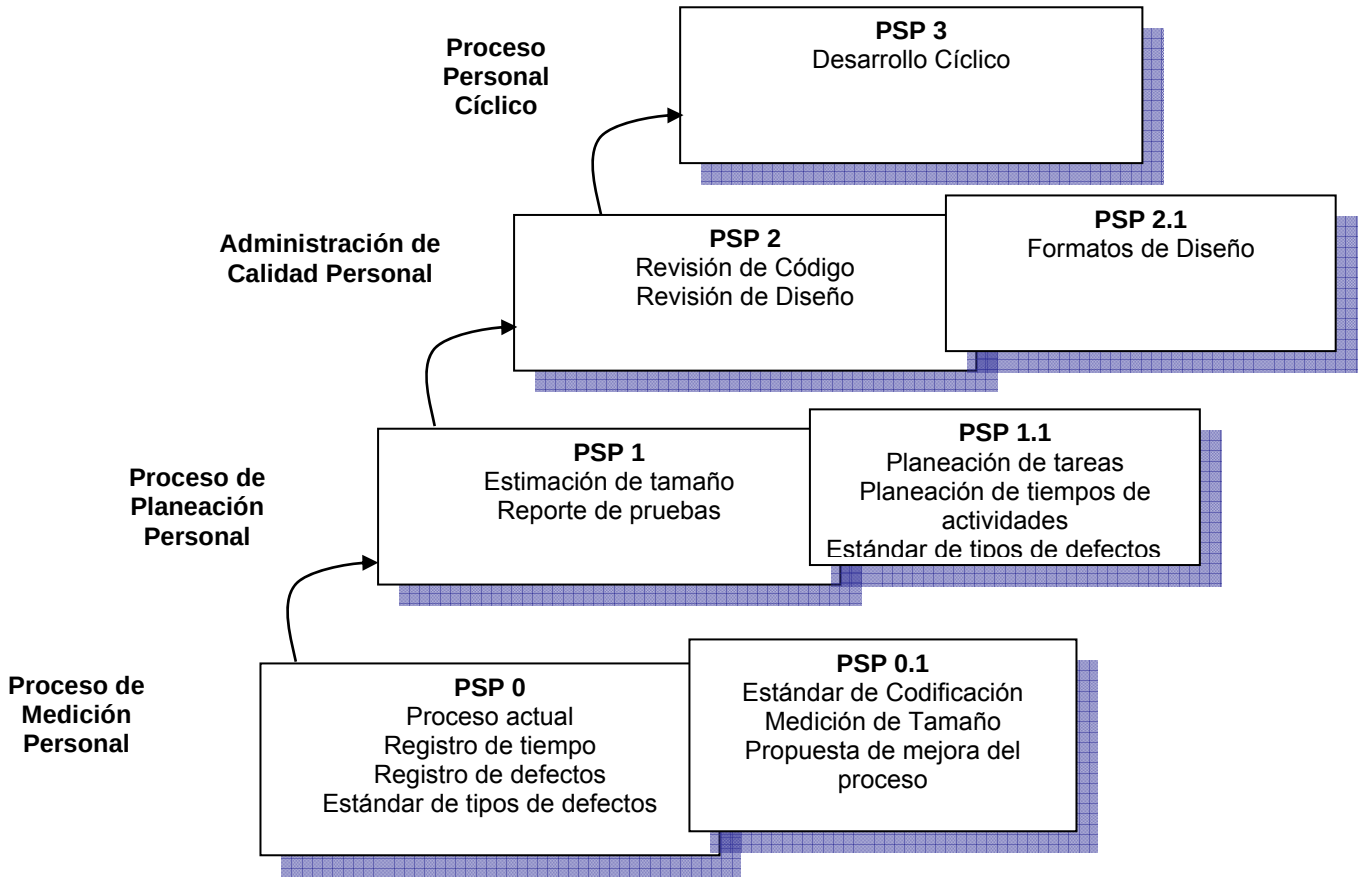


Figura 11: Niveles del PSP

ANEXO 2 PLAN DEL PROYECTO.

Resumen del plan del proyecto del PSP.

Estudiante _____ Fecha _____

Programa _____ Programa # _____

Profesor _____ Lenguaje _____

Resumen	Plan	Real	Hasta la Fecha
Minutos/LOC			
LOC/Hora			
Defectos/KLOC			
Rendimiento			
V/F			
Tamaño Programa (LOC):			
Total Nuevo & Cambiado			
Tamaño Máximo			
Tamaño Mínimo			

Tiempo por Fase (min.)	Plan	Real	Hasta la Fecha	% Hasta la Fecha
Planificación				
Diseño				
Codificación				
Revisión del código				
Compilación				
Pruebas				
Postmorten				
Total				
Tiempo Máximo				
Tiempo Mínimo				

Defectos introducidos	*Plan*	*Real*	*Hasta la Fecha*	*% Hasta la Fecha*	*Def./Hora*
Planificación					
Diseño					
Codificación					
Revisión del código					
Compilación					
Pruebas					
Total					

Defectos eliminados	*Plan*	*Actual*	*Hasta la Fecha*	*% Hasta la Fecha*	*Def./Hora*
Planificación					
Diseño					
Codificación					
Revisión del código					
Compilación					
Pruebas					
Total					

Tabla 29: Plan del proyecto.

La interpretación de cada uno de los elementos es la siguiente:

Encabezamiento

- Estudiante: Nombre del estudiante.

- Fecha: Fecha del momento en que se emite el resumen del proyecto.
- Programa: Nombre del programa.
- Programa #: Numero del programa.
- Lenguaje: Lenguaje que se utilizara para escribir el programa.
- Profesor: Nombre del profesor.

Resumen

- Minutos/LOC

Plan: Escribe los Minutos/LOC planificados para este proyecto.

Real: Divide el tiempo total de desarrollo (real) por el tamaño real del programa.

Hasta la fecha:

- LOC/Hora

Plan: Calcula las LOC por hora planificada para este programa dividiendo 60 por los Minutos/LOC de la casilla de Plan.

Real: Para LOC/Hora Real divide 60 por los Minutos/LOC Reales.

Hasta la Fecha:

- Defectos/KLOC

Plan: Encuentra los Defectos/KLOC Hasta la Fecha del programa más reciente. Utilizalo como los Defectos/KLOC planificados.

Real: $\frac{1000^* \text{ Total de defectos reales}}{\text{LOC Total Nuevas \& Cambiadas Reales}}$

Hasta la Fecha: $\frac{1000^* \text{ Total de defectos Hasta la Fecha}}{\text{LOC Total Nuevas \& Cambiadas Hasta la Fecha}}$

- Rendimiento

Plan: $\frac{100*(\text{número de defectos eliminados antes de la compilación})}{\text{Número de defectos introducidos antes de la compilación}}$

Real: $\frac{100*(\text{número de defectos eliminados antes de la compilación})}{\text{Número de defectos introducidos antes de la compilación}}$

Hasta la Fecha: $\frac{100*(\text{número de defectos eliminados antes de la compilación})}{\text{Número de defectos introducidos antes de la compilación}}$

- V/F

Plan: $\frac{\text{Tiempo de revisión de código planificado}}{\text{Tiempo de compilación (P)+Tiempo de prueba (P)}}$

P: Planificado.

Real: $\frac{\text{Tiempo de revisión de código real}}{\text{Tiempo de compilación (R)+Tiempo de prueba (R)}}$

R: Real

Hasta la Fecha: $\frac{\text{Tiempo de revisión de código Hasta la Fecha}}{\text{Tiempo de compilación (H)+Tiempo de prueba (H)}}$

H: Hasta la Fecha

Tamaño Programa (LOC)

- Total Nuevo & Cambiado

Plan: Valor estimado de Total Nuevas & Cambiadas

Real: Cuenta y escribe las LOC Nuevas & Cambiadas Reales.

Hasta la Fecha: Añade LOC Reales Nuevas & Cambiadas a las LOC Hasta la Fecha Nuevas & Cambiadas de programas anteriores.

- Tamaño Máximo

Plan: Valor estimado de Tamaño Máximo.

- Tamaño Mínimo

Plan: Valor estimado de Tamaño Mínimo.

Tiempo por fase (min)

Plan: Del Resumen del Plan de Proyecto de un programa reciente, busca los valores de % Hasta la Fecha para cada fase. Utilizando el % Hasta la Fecha de programas anteriores calcula el tiempo planificado para cada fase.

Real: Una vez acabado el trabajo, anota el tiempo real en minutos que has gastado en cada fase de desarrollo.

Hasta la Fecha: Para cada fase, escribe la suma del tiempo real y el tiempo Hasta la Fecha de los programas más recientes.

% Hasta la Fecha: Para cada fase, escribe 100 multiplicado por el tiempo Hasta la Fecha y lo divides por el total del tiempo Hasta la Fecha

Para el tiempo total de desarrollo, multiplica el valor de las LOC Total Nuevas & Cambiadas por Minutos/LOC.

Para el Tiempo Máximo, multiplica el tamaño máximo por los Minutos/LOC.

Para el Tiempo Mínimo, multiplica el tamaño mínimo por los Minutos/LOC.

Defectos introducidos

Plan: Estima el número total de defectos a introducir en el programa.

$$\text{Total de defectos planificados} = \frac{\text{Defectos/KLOC planificados} * \text{LOC Total Nuevas \& Cambiadas}}{1000}$$

Real: Se localiza y anota el número real de defectos introducidos en cada fase.

Hasta la Fecha: Para cada fase, anota la suma de los defectos reales y los defectos Hasta la Fecha de los programas mas recientes.

% Hasta la Fecha: Para cada fase

$$\frac{100 * \text{Defectos Hasta la Fecha (para esa fase)}}{\text{Total de Defectos Hasta la Fecha}}$$

Defectos/hora: Calcula los defectos introducidos por hora para la fase de diseño y codificación.

$$\frac{60^* \text{ Defecto Hasta la Fecha (diseño o codificación)}}{\text{Tiempo Hasta la Fecha (diseño o codificación)}}$$

Defectos eliminados

Plan: En la fila de total, anota el total de defectos estimados. Utiliza los valores de % Hasta la Fecha de los programas mas recientes, calcula los defectos eliminados planificados para cada fase.

Real: Se localiza y anota el número real de defectos eliminados en cada fase.

Hasta la Fecha: Para cada fase, anota la suma de los defectos reales y los defectos Hasta la Fecha de los programas mas recientes.

% Hasta la Fecha: Para cada fase

$$\frac{100^* \text{ Defectos Hasta la Fecha (para esa fase)}}{\text{Total de Defectos Hasta la Fecha}}$$

Defectos/hora: Calcula los defectos eliminados por hora para la fase de revisión de código, compilación y prueba.

$$\frac{60^* \text{ Defecto Hasta la Fecha (revisión de código, compilación o prueba)}}{\text{Tiempo Hasta la Fecha (revisión de código, compilación o prueba)}}$$

ANEXO 3 EJEMPLO DE MÉTRICAS EN EL TSP.

Error de estimación de tamaño.

La estimación es una de las actividades prioritarias en la gestión de proyectos, por lo que es importante estimar el tamaño basado en datos históricos, para esto se utiliza:

Error Estimación Tamaño =100* (Tamaño Real–Tamaño Estimado)/Tamaño Estimado

Con los datos que se obtienen podremos deducir que a medida que los equipos van adquiriendo mayor experiencia mejoran la estimación.

Densidad de defectos.

Para producir programas libres de defectos es importante medir la calidad de los programas, y con esta información dar los pasos para mejorar. El reto que tenemos es llegar a reducir los defectos cada vez más, alcanzando el valor lo más próximo a cero. Un defecto es cualquier cosa que reduce la capacidad del software para cumplir de manera completa y eficiente la necesidad de los usuarios. El defecto es algo objetivo que se puede identificar, describir y contabilizar.

La construcción de software es un proceso que se puede evaluar a lo largo de las etapas de los productos intermedios. Una alta densidad de defectos en los productos intermedios se traduce en un esfuerzo añadido a causa del trabajo que hay que volver a realizar o el tiempo incurrido en solucionar defectos. Sin embargo, esta situación puede mejorar si se lleva un registro de los defectos que se están produciendo y si se identifican los factores que los ocasionan. La densidad de defectos es un indicador temprano de la calidad final y se define mediante la siguiente formula:

Densidad defectos=Defectos eliminados /KLOC

A medida que los equipos adquirieron mayor experiencia, así como la introducción de revisiones de diseño y código, se produce una disminución de los defectos inyectados, con lo que se obtiene un producto de mayor calidad. La reducción de la densidad de defectos total se traduce directamente en una reducción en la cantidad de trabajo que hay que rehacer y minimización de posibles retrasos y rechazo de parte del cliente.

Productividad

Una métrica importante en toda organización es el índice de productividad, es decir, la generación de producto por unidad de tiempo. Se puede decir que “a medida que se avanza en los ciclos de desarrollo del proyecto, la productividad se incrementa”. La productividad está definida por la relación que existe entre la cantidad de líneas de código generadas por unidad de tiempo o esfuerzo realizado.

Productividad = LOC/Horas

Costes

La relación coste y productividad es otra inquietud en los proyectos software y se debe responder a la pregunta si están teniendo alta productividad pero a costes elevados. Entonces, es importante medir si realmente el indicador de productividad alto permite reducir costes. Los proyectos, a medida que van mejorando, no sólo la productividad se debe incrementar, sino que se deben reducir significativamente los costes”. La relación que existe entre los costes del proyecto por líneas de código generadas se define mediante la siguiente formula:

Productividad (Euros/LOC)=Coste proyecto/LOC

Coste proyecto= X Euros* hora de trabajo.

ANEXO 4 CMM

	Características	KPAs	Resultado
5 Optimizado	Bases cualitativas para una inversión capital continua en la automatización y mejoramiento de la retroalimentación de los procesos.	-Prevención de defectos -Administración de la tecnología de cambio -Administración del proceso de cambio	Productividad y calidad
4 Administrad	(cuantitativo) -Procesos medidos -Control estadístico razonable sobre la calidad del producto	-Medición y análisis de los procesos -Administración de la calidad	
3 Definido	(cuantitativo) -Costo y planificación confiables -Procesos definidos e institucionalizados -Rendimiento de la calidad mejorado pero impredecible	-Definición y mejoramiento de los -procesos organizacionales -Programas de entrenamiento -Administración integrada de software -Coordinación intergrupal -Inspecciones entre compañeros -Ingeniería del producto de software	
2 Repetible	(intuitivo) -Procesos dependientes de los individuos - Costo y calidad altamente variables - Métodos y procedimientos informales y "ad hocs"	-Administración de los requerimientos -Planificación y vigilancia de los proyectos de software -Administración de la subcontratación de software -Aseguramiento de la calidad del software -Administración de la configuración de software	Riesgos
1 Inicial	-Ad hoc -Caótico -costos, planificación y rendimiento de la calidad impredecibles		

Figura 12: Modelo de madurez de las capacidades (CMM).

Nivel 1: Inicial. Este nivel es el primer estado en la evolución de las organizaciones que desarrollan software. En este nivel se encuentran todas las empresas que no han logrado implementar las prácticas básicas de administración de proyectos e ingeniería de software definidas a partir del nivel 2 o superiores.

Nivel 2: Repetible. En este nivel se establecen prácticas básicas de administración de proyecto que permiten establecer control de requerimientos, calendarios y costos. El equipo que desarrollo el proyecto puede aprovechar su experiencia e inversión para aplicarla en un nuevo proyecto. A partir de este nivel es importante llevar una implementación controlada de métricas y medidas de software ya que si no se hace así desde los primeros niveles, en los próximos será muy tardado llevar a cabo una transición.

Nivel 3: Definido. En este nivel la empresa ha definido un conjunto de procesos, metodologías y herramientas usados por todos lo niveles involucrados en el proceso, tanto administrativos como desarrolladores. Existen pautas y criterios definidos para adaptar un proceso estándar a las necesidades y características propias de cada proyecto. En el proceso ya no existen características individuales, pues todos conocen el proceso.

Nivel 4: Administrativo. En este nivel la organización mide la calidad del producto y del proceso de software. Ambas partes son seguidas en forma cuantitativa y se controlan mediante métricas detalladas. En este nivel el proyecto se entiende como un conjunto de procesos y la administración de los mismos y sus cambios. Todos los procedimientos y métricas necesarios en los niveles anteriores se integran en este nivel.

Nivel 5: Optimo. La característica principal aquí es que la organización entera lleva a cabo mejoras continuas en el proceso, en base de retroalimentación cuantitativa y al ensayo de ideas y tecnologías innovadoras. Todos los cambios son vigilados y controlados con el sistema de métricas.

Key Process Areas(KPA)

Para cada nivel (excepto el 1), el CMM especifica algunas KPA, que representan las áreas en las que una organización debe enfocarse si desea alcanzar algún nivel en particular.

Las KPA para los niveles de madurez existentes pueden ser usadas para verificar la capacidad del proceso actual e identificar las áreas que necesitan ser reforzadas para alcanzar el siguiente nivel. (María Teresa Ventura Miranda. tere@exodus.dcaa.unam.mx) Septiembre de 2002

ANEXO 5 EJEMPLO DE MÉTRICAS EN EL CMM PARA LAS REVISIONES.

Efectividad de eliminar los defectos en una Revisión:

$$EED = \frac{DE_i}{DL} * 100$$

$$DL = DE_i + DEP$$

$$DEP = \sum_{k=i+1}^{n} DE_k$$

DE: cantidad de defectos detectados durante la revisión,

DEP: cantidad de defectos encontrados posterior a la revisión, es decir la cantidad de defecto encontrados en la n-i restantes revisiones que se indican en el plan de revisiones y auditorias del proyecto. También puede calcularse este valor considerando los defectos detectados en las revisiones efectuadas hasta el momento (k = m, con i < m < n) en que se desea analizar la métrica, pero serán resultados parciales que pueden cambiar al finalizar el producto.

DL: cantidad total de defectos presentes en el producto, cuando éste ha sido terminado y se entrega al cliente para su operación.

Ahora bien, esta medida da una vista global de la efectividad de la revisión, pero en ocasiones no basta con esta información y es necesario profundizar para conocer cuáles tipos de errores no han sido detectados y que por tanto conspiran contra la efectividad de dicha Revisión, para ello se propone la métrica siguiente.

Efectividad de eliminar los defectos de la fase j en la revisión i.

Permite a los directivos conocer la efectividad de las revisiones en cuanto a la cantidad de defectos que pertenecen a una fase y que son encontrados oportunamente. Por tanto, se puede analizar la cantidad de defectos, por ejemplo de la fase de requisitos, que se han propagado hasta la implantación del sistema o hasta cualquier otra fase de desarrollo del proyecto.

$$EED_{i,j} = \frac{DE_{i,j}}{DL_j} * 100$$

DEi,j: cantidad de defectos detectados durante la revisión i, correspondientes a la fase j.

DLj: cantidad total de defectos presentes en el producto correspondientes a la fase j.

El comportamiento de estas dos primeras métricas puede ser representado en un único modelo que de una idea global de la eficiencia de las Revisiones y de cada una de las fases consideradas en ellas. Por ejemplo, se pudiera plotear los valores de la efectividad de diferentes revisiones realizadas a un proyecto y compararlas entre sí.

Densidad de defectos.

$$DD = \frac{DT}{TP}$$

DT: cantidad total de defectos encontrados en el producto.

TP: tamaño del producto, puede ser estimado en líneas de código (en miles) o en puntos de función.

Da una medida de la proporción de defectos del producto con respecto a su tamaño. De la forma en que está definida debe ser evaluada al finalizar el producto y puede ser aprovechada como experiencia para proyectos futuros, no obstante en las etapas tempranas de desarrollo del proyecto esta métrica puede ser utilizada considerando la estimación preliminar de líneas de código o puntos de función que haya sido considerada en la planificación del proyecto. La densidad de defectos de cada uno de los proyectos revisados en la empresa da una idea de la calidad de los productos que se desarrollan en esta y de la mejora continua del proceso de revisiones al transcurrir un periodo de tiempo determinado.

ANEXO 6. CALIDAD DEL PRODUCTO. ISO 9126.

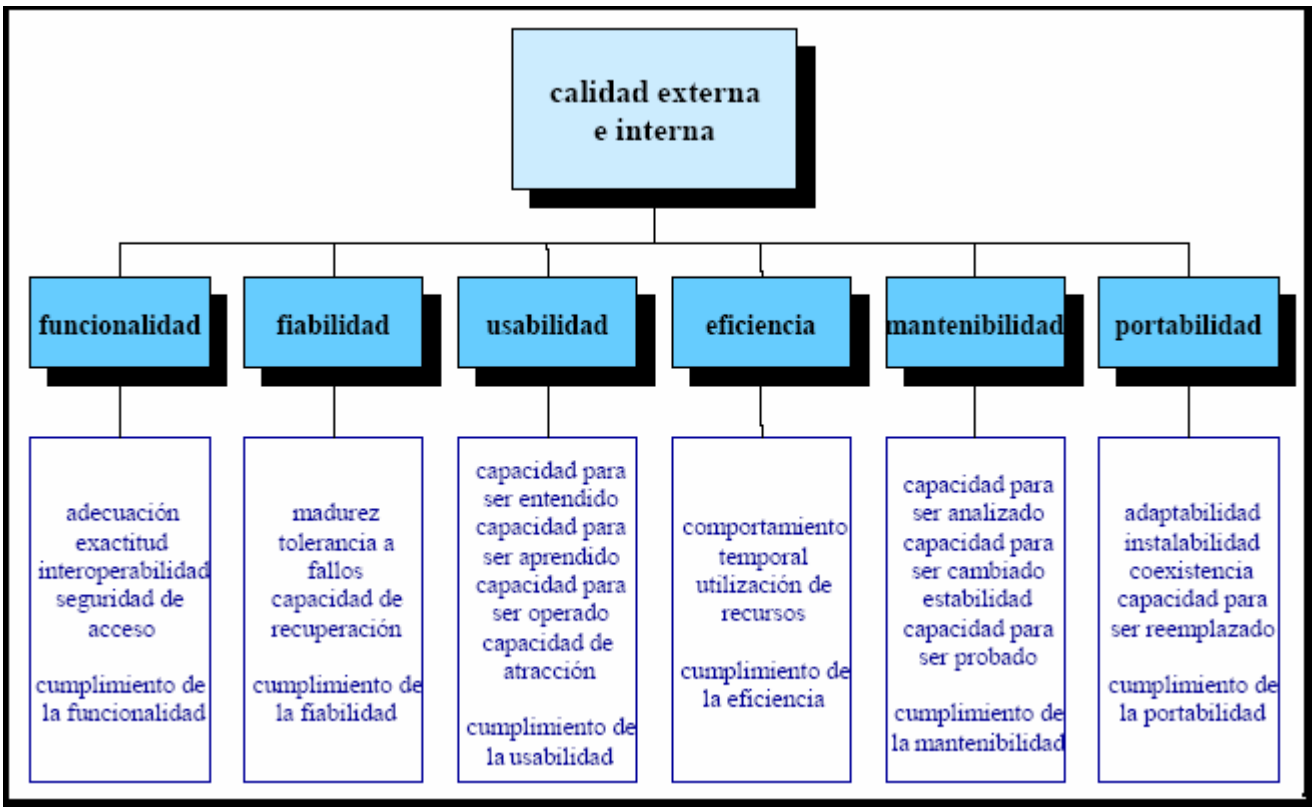


Figura 13: Características de la ISO/IEC 9126.

Funcionalidad

- Adecuación: Capacidad del producto software para proporcionar un conjunto apropiado de funciones para tareas y objetivos de usuario especificados.
- Exactitud: Capacidad del producto software para proporcionar los resultados o efectos correctos o acordados, con el grado necesario de precisión.
- Interoperabilidad: Capacidad del producto software para interactuar con uno o más sistemas especificados.
- Seguridad de acceso: Capacidad del producto software para proteger información y datos de manera que las personas o sistemas no autorizados no puedan leerlos o modificarlos, al tiempo que no se deniega el acceso a las personas o sistemas autorizados.

- Cumplimiento funcional: Capacidad del producto software para adherirse a normas, convenciones o regulaciones en leyes y prescripciones similares relacionadas con funcionalidad.

Fiabilidad

- Madurez: Capacidad del producto software para evitar fallar como resultado de fallos en el software.
- Tolerancia a fallos: Capacidad del software para mantener un nivel especificado de prestaciones en caso de fallos software o de infringir sus interfaces especificados.
- Capacidad de recuperación: Capacidad del producto software para reestablecer un nivel de prestaciones especificado y de recuperar los datos directamente afectados en caso de fallo.
- Cumplimiento de la fiabilidad: Capacidad del producto software para adherirse a normas, convenciones o regulaciones relacionadas con al fiabilidad.

Usabilidad

- Capacidad para ser entendido: Capacidad del producto software que permite al usuario entender si el software es adecuado y cómo puede ser usado para unas tareas o condiciones de uso particulares.
- Capacidad para ser aprendido: Capacidad del producto software que permite al usuario aprender sobre su aplicación.
- Capacidad para ser operado: Capacidad del producto software que permite al usuario operarlo y controlarlo.
- Capacidad de atracción: Capacidad del producto software para ser atractivo al usuario.
- Cumplimiento de la usabilidad: Capacidad del producto software para adherirse a normas, convenciones, guías de estilo o regulaciones relacionadas con la usabilidad.

Eficiencia

- Comportamiento frente al tiempo: Capacidad del producto software para proporcionar una respuesta y un tiempo de procesamiento apropiados al desarrollar sus funciones bajo condiciones establecidas.

- Uso de recursos: Capacidad del producto software para utilizar un apropiado numero de recursos y tiempo de ejecución cuando el software desarrolla sus funciones bajo condiciones establecidas.
- Cumplimiento de la eficiencia: Capacidad del software relacionada con el grado de conformidad con estándares, convenciones o regulaciones existentes en leyes o prescripciones similares.

Mantenibilidad

- Capacidad para ser analizado: Es la capacidad del producto software para serle diagnosticadas deficiencias o causas de los fallos en el software, o para identificar las partes que han de ser modificadas.
- Capacidad para ser cambiado: Capacidad del producto software que permite que una determinada modificación sea implementada.
- Estabilidad: Capacidad del producto software para evitar efectos inesperados debidos a modificaciones del software.
- Capacidad para ser probado: Capacidad del producto software que permite que el software modificado sea validado.
- Cumplimiento de la mantenibilidad: Capacidad del producto software para adherirse a normas o convenciones relacionadas con la mantenibilidad.

Portabilidad

- Adaptabilidad: Capacidad del producto software para ser adaptado a diferentes entornos especificados, sin aplicar acciones o mecanismos distintos de aquellos proporcionados para este propósito por el propio software considerado.
- Instalabilidad: Capacidad del producto software para ser instalado en un entorno especificado.
- Coexistencia: Capacidad del producto software para coexistir con otro software independiente, en un entorno común, compartiendo recursos comunes.
- Capacidad para reemplazar: Capacidad del producto software para ser usado en lugar de otro producto software, para el mismo propósito, en el mismo entorno.

- Cumplimiento de la portabilidad: Capacidad del producto software para adherirse a normas o convenciones relacionadas con la portabilidad.

ANEXO 7. MODELO SISTÉMICO DE CALIDAD (MOSCA). NIVELES DE MOSCA.

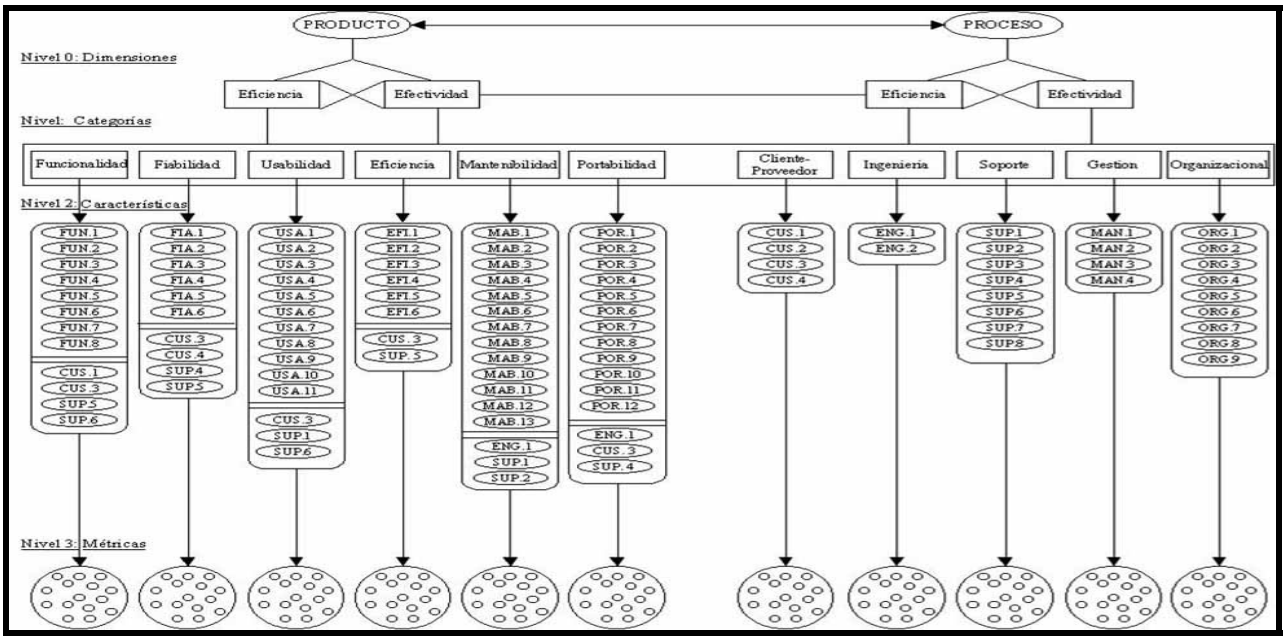


Figura 14: Modelo Sistémico de Calidad (MOSCA). Niveles de MOSCA.

Nivel 0: Dimensiones. Eficiencia del proceso, Efectividad del proceso, Eficiencia del producto y Efectividad del producto son las cuatro dimensiones propuestas en el prototipo de modelo. Sólo un balance y una buena interrelación entre ellas permiten garantizar la calidad Sistémica global de una organización.

Nivel 1: Categorías. Se contemplan 11 categorías: 6 pertenecientes al producto y las otras 5 al proceso de desarrollo.

Producto: Funcionalidad (FUN), Fiabilidad (FIA), Usabilidad (USA), Eficiencia (EFI), Mantenibilidad (MAB) y Portabilidad (POR).

Proceso: Cliente-Proveedor (CUS), Ingeniería (ENG), Soporte (SUP), Gestión (MAN) y Organizacional (ORG).

Nivel 2: Características. Cada categoría tiene asociado un conjunto de características (56 asociadas al producto y 27 al proceso de desarrollo), las cuales definen las áreas claves a satisfacer para lograr, asegurar y controlar la calidad tanto en el producto como en el proceso.

Nivel 3: Métricas. La cantidad de métricas asociadas a cada una de las características que conforman MOSCA es de 587 en total.

ANEXO 8. DIAGRAMA DE ACTIVIDAD. PROCESO PARA EVALUAR EL SOFTWARE EDUCATIVO.

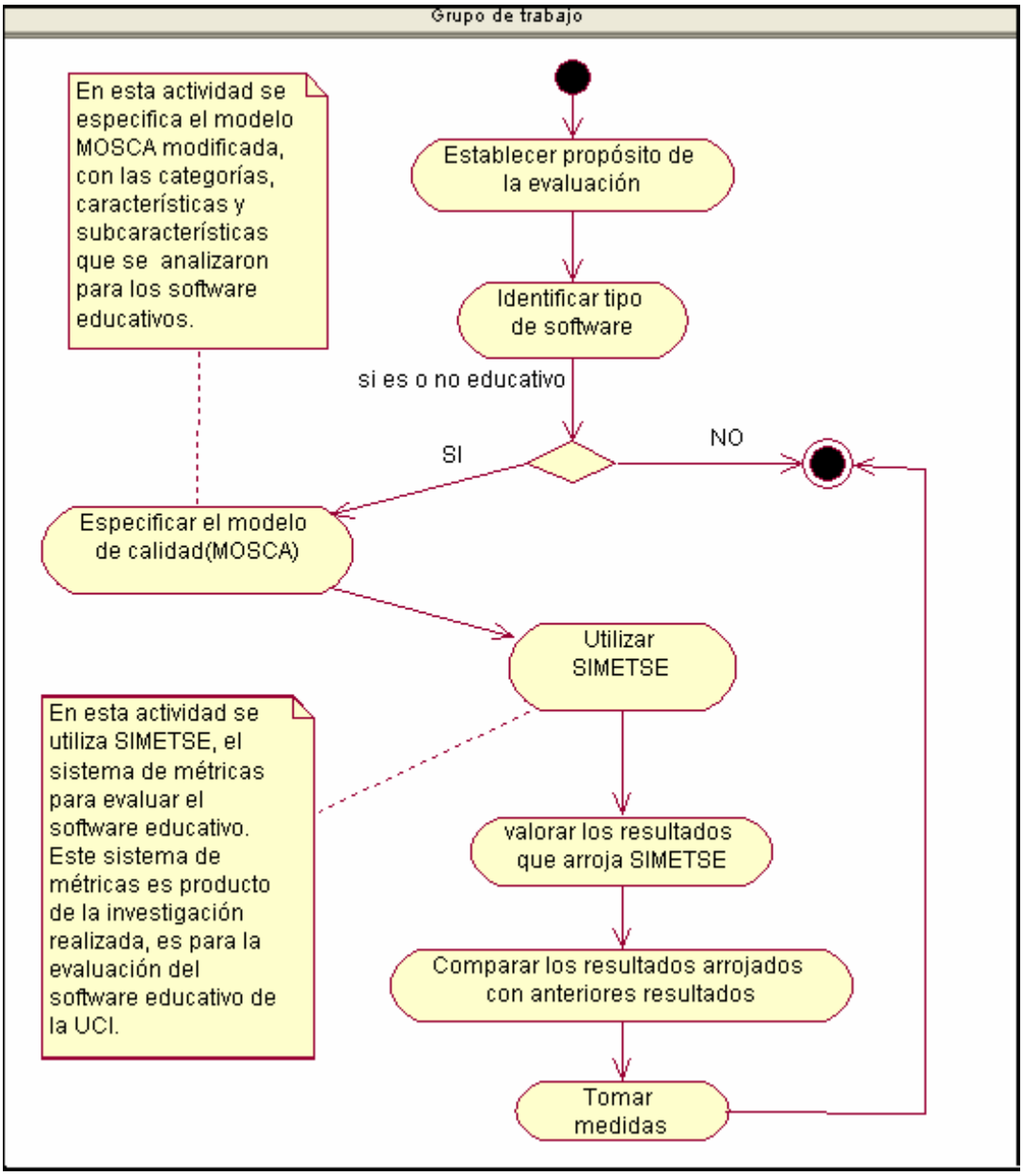


Figura 15: Diagrama de actividades: Proceso para evaluar el software educativo.

ANEXO 9. ENCUESTA REALIZADA A LOS PROYECTOS QUE DESARROLLAN SOFTWARE EDUCATIVO EN LA UCI.

Nombre:

Proyecto:

Facultad:

1. **¿Cómo controlan la calidad del software?**

Pruebas_____ Métricas_____ No se controla_____

Otros_____

En caso que sea otros, mencionarlos_____

2. **¿Qué riesgos tiene una mala calidad?**

_____

3. **¿Tienen algún conocimiento de las métricas que se utilizan para la evaluación del software?**

Si____ NO_____

4. **¿Aplican dichas métricas para la evaluación de software educativo en la UCI?**

Si____ No____

En caso positivo: ¿Cuáles métricas aplican?_____

¿Y en qué etapas del proyecto aplican las métricas?_____

En caso negativo ¿por qué no?_____

5. **¿Es importante la aplicación de las métricas para la evaluación del software educativo?**

Sí___ NO_____

¿Por qué? _____

6. **¿Las métricas son una alternativa para eliminar los errores en los productos?**

Sí_____ No_____

¿Por qué?_____

7. ¿Ayudan las métricas a mejorar la calidad del software educativo?

Sí_____ No_____

8. Entre lo niveles organizacionales están:

PSP____ TSP_____ CMM_____

Marque con una cruz el que utilizan

¿Qué importancia le dan al nivel organizacional utilizado?

Alta____ Media_____ Baja_____

¿Por qué?_____

Nota:

(CMM). Capability Maturity Model
(PSP). Personal Software Process
(TSP). Team Software Process.

ANEXO 10. ENTREVISTA REALIZADA A EXPERTOS EN EL TEMA.

Cuestionario sobre la aplicación de las métricas para la evaluación de software educativo en la UCI.

Nombre:

Lugar en el cual trabaja:

Cargo que ocupa:

Facultad:

Preguntas:

1. ¿Cuál es la situación del software educativo en la UCI? (Si la productividad es alta, media o baja y el por qué)

2. ¿Cómo se evalúa el software educativo en la UCI?

3. ¿Qué importancia le dan a la calidad?

4. ¿Qué calidad tienen los productos de software educativo que se desarrollan en la UCI?

5. ¿Sufren atrasos los proyectos realizados en la UCI? ¿Por qué?

6. ¿Conoce si se utilizan métricas para evaluar el software educativo en la UCI? ¿Cuáles métricas se utilizan?

En caso de que no se aplique las métricas

1. ¿No afectan a los proyectos la no aplicación de métricas? ¿En qué modo se afectan? ¿Por qué no se aplican?

2. ¿Cuáles métricas se podrían aplicar? ¿Cuáles serían más fáciles de usar según su conocimiento o gusto?

GLOSARIO DE TÉRMINOS.

A continuación se presenta por orden alfabético algunos de los términos utilizados en la realización de este documento y que puedan crear confusión en la comprensión del mismo.

A

Atributo: Determinada característica que pretendemos medir y cuantificar. Ejemplo: esfuerzo, fiabilidad, mantenibilidad, calidad y tamaño.

C

CMM: Modelo de Madurez de la Capacidad del Desarrollo del Software o Capability Maturity Model. Ambiente de trabajo de madurez de procesos de software desarrollado por el Software Engineering Institute (SEI) de la Universidad de Carnegie-Mellon.

CMMI: Modelo de capacidad de madurez integrada.

D

Defecto: indicadores de que un artefacto no funciona como ha sido especificado, o cualquier otra característica indeseable. Cualquier requerimiento, elemento de diseño o de implementación que si no es cambiado, causará un diseño, implementación, prueba, uso, o mantenimiento inapropiado del producto.

I

ISO: Organización Internacional para la Estandarización o International Standard Organization.

IEEE: Instituto de Ingenieros Eléctricos y Electrónicos o The Institute of Electrical and Electronics Engineer.

IP: Infraestructura productiva.

K

KPAs: Key Process Areas.

M

Métrica: asignación de valor a un atributo de una entidad propia del *software*, ya sea un producto o un proceso.

Métricas del software: es una medida cuantitativa del grado en el que un sistema, un componente o un proceso poseen un determinado atributo.

MOSCA: Modelo Sistémico de Calidad.

MEPS: Mejora de Estadística del Proceso del Software.

N

Norma o estandar: documento aprobado por consenso por un organismo reconocido, que proporciona reglas, pautas y/o características para uso común, con el objeto de obtener un óptimo nivel de resultados en un contexto dado.

P

Producto: nos referimos, por ejemplo, a un código, un diseño, una specificación, etc.

PSP: Proceso Personal de Software o Personal Software Process. Proceso de automejoramiento, basado en CMM, diseñado para ayudar a controlar, administrar y mejorar la forma en que se trabaja individualmente.

Proceso: etapa de la construcción del software y a la manera de llevarlo a cabo: un diseño, unas pruebas, etc.

S

Software educativo: cualquier producto basado en computadora con una finalidad educativa.

SEI: Instituto de ingeniería del software.

SIMETSE: Sistema de métricas para evaluar el software educativo.

T

TSP: Proceso de Software del Equipo o Team Software Process. Proceso, basado en CMM, diseñado para ayudar a controlar, administrar y mejorar la forma en que trabaja un equipo de desarrollo de software.

U

UCI: Universidad de las Ciencias Informáticas.