Universidad de las Ciencias Informáticas

Facultad 8

Título: Propuesta de métricas para la estimación de proyectos

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autora: Mailyn Tapanes Alfonso

Tutoras: Ing.Dayami Rodríguez Brito

Ing.Yunexis Rodríguez Baryolo

Ciudad de La Habana, julio 2008

´´Año 50 de la Revolución´´

Declaración de Autoría

Declaro ser autora de este trabajo y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

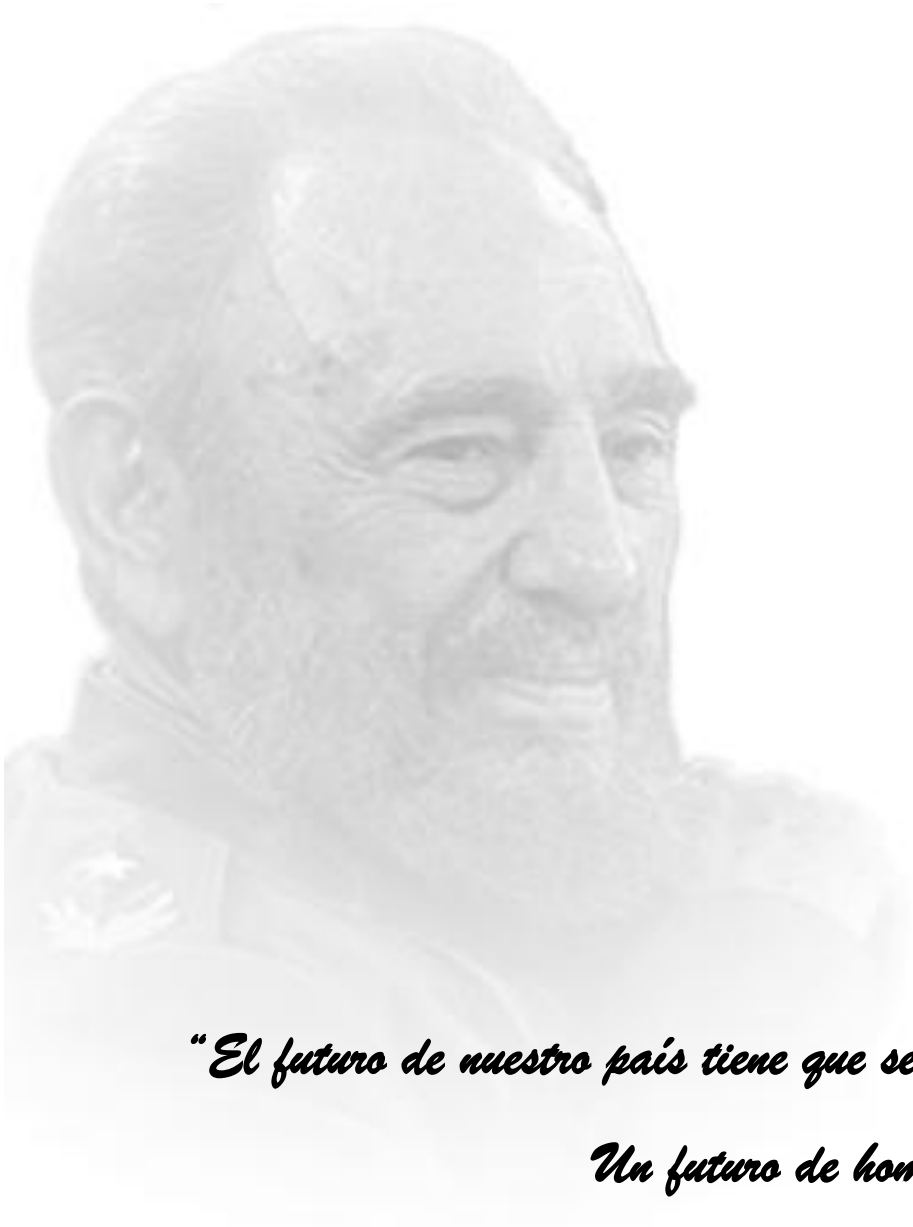
Para que así conste firmo la presente a los ___ días del mes de Julio de 2008.

_____

Firma de la Autora

_____ _____

Firma de la Tutora Firma de la Tutora



"El futuro de nuestro país tiene que ser necesariamente,

Un futuro de hombres de ciencia".

Fidel Castro Ruz

Agradecimientos

A mi abuela Dalia y a mi mamá por todo su amor, preocupación y por todo el esfuerzo que han hecho por mí.

A mi tía por ser como una madre para mí, y el apoyo que siempre me ha brindado.

A mi papá por su preocupación, apoyo y por darme todo lo que he necesitado en todos estos años.

A mis hermanas, especialmente a Marialita por quererme tanto y compartirlo todo conmigo.

A mi abuela Haydee y a mis abuelos por siempre ayudarme y preocuparse por mí.

A Yasely porque a pesar de que el destino nos separó y hoy no es mi compañera de tesis nunca me abandonó y me siguió ayudando en todo momento. Por muchas palabras lindas que pusiera aquí no tengo como agradecerle todo lo que hizo por mí.Mil gracias Yase.

A Nidia, por estar siempre a mi lado en los buenos y malos momentos, por ser la única compañera en estos cinco años con la que siempre he podido contar y confiar, gracias mi verdadera amiga.

A todos mis compañeros que de una forma u otra me brindaron su ayuda cuando la necesité, en especial a Dixson, Marcel, Yeney, Dianelys.

A mis tutoras, en especial a Dayami por brindarme toda su ayuda, su disposición incondicional y por dedicarme tantas horas de su tiempo.

Al Prof. Yescarles por ayudarme y preocuparse por mí en estos cinco años de mi carrera.

A la UCI, a todos mis profesores y a nuestro Comandante en Jefe por darme la oportunidad de estudiar y graduarme hoy como Ingeniera en Ciencias Informáticas.

Dedicatoria

A mi abuela Dalia por estar siempre a mi lado,

Por tantos y tantos sacrificios que ha hecho por mí,

Por sus desvelos cuando tenía algún problema,

Por todas sus oraciones cuando tenía pruebas

Y por regalarme todo el amor del mundo.

Resumen

La calidad es el término más importante en cualquier empresa productora de software. Los proyectos de software para que sean exitosos deben partir de una buena planificación, para esto es necesario realizar estimaciones realistas y para realizar buenas estimaciones se necesitan umbrales de mediciones. Las mediciones se obtienen luego de aplicar métricas ya sean básicas o derivadas y a partir de esto la importancia que se le tributa a la aplicación de las métricas. En la Universidad de las Ciencias Informáticas (UCI), aún después de los 6 años de creada; el proceso de medición no ha madurado como es debido, partiendo de que no se ha creado una cultura de medición en el estudiantado que es la mayor fuerza de trabajo. Evidentemente ya que no se mide eficientemente no se puede contar con registros históricos para poder realizar mejores estimaciones y mejorar el proceso productivo en cuanto a tiempo, costo y el esfuerzo requerido. En la presente tesis se realiza un reconocimiento y estudio del estado actual de las métricas de software existentes, tanto a nivel mundial, nacional, así como en la universidad, con vistas a realizar una propuesta de aplicación de las métricas que más se adecúen en estos momentos teniendo en cuenta las características propias de la institución.

Palabras Claves

Calidad de Software, planificación, estimación, métricas.

Índice

Índice de Figuras

Índice de Tablas

Introducción

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.

De manera 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.

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 en paralelo desde la planificación del producto hasta la fase de producción del mismo.

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 la calidad. Entonces, se especifican las métricas, que estas proveen la información necesaria para la toma de decisiones técnicas. Vale aclarar que las métricas no son estándares.

Considerando la importancia de obtener productos con la calidad requerida en los proyectos de software, cabe preguntarse, ¿Qué papel juega la planificación en el alcance de la calidad de un producto?, ¿Hasta que punto puede ser importante planificar de forma eficiente un proyecto de software?, ¿Cómo poder planificar?

Todo proyecto de ingeniería de software debe partir con un buen plan, pero lamentablemente, la planificación es una tarea nada trivial. Uno de los aspectos que dificultan la labor de administradores y jefes de proyecto en torno a la planificación es la difícil tarea de realizar buenas estimaciones de costos y plazos realistas porque no se cuentan con la información requerida, ni se mide tan siquiera.

La medida del software es un mecanismo muy útil para gestionar proyectos de software y para mejorar la calidad del producto obtenido, otras de las razones que se tiene para medir el software son indicar esa calidad del software que se pretende continuamente, evaluar la productividad de las personas que desarrollan el software, evaluar los beneficios en términos de productividad, derivados del uso de

diversos métodos y herramientas de la Ingeniería de Software, establecer una línea base para la estimación, y así ayudar a justificar el uso de nuevas herramientas.

Para realizar la estimación del esfuerzo de un proyecto de software se debe disponer de información histórica que proporcione una base de partida. Cada proyecto se define mediante una serie de atributos, tales como puntos de función, esfuerzo de trabajo, tipo de proyecto, tiempo de desarrollo, número de integrantes del equipo de trabajo, si se ha utilizado una metodología. Lo que se persigue es aprender una cierta función de manera que conocidos una serie de atributos que posibiliten la obtención de buenas medidas que ayuden a realizar mejores estimaciones de los proyectos de software a desarrollar.

La Universidad de las Ciencias Informáticas (UCI) constituye un nuevo modelo de formación-investigación- producción en el campo de las TIC.

Ofrece amplias posibilidades al desarrollo de la Industria Cubana del Software y los servicios informáticos y su impacto se hace sentir ya en diferentes sectores de la sociedad y la economía nacional.

La UCI está creada sobre la base del nuevo concepto de universidad productiva, logrando una fuerte vinculación Universidad-Empresa. En la UCI la producción es un problema social, político y económico, el cien por ciento de los estudiantes y profesores deben vincularse a la producción participando en proyectos de alto valor tanto para el mercado nacional, como internacional, se plasma la concepción de que la docencia se realice desde la producción.

En la Universidad de las Ciencias Informáticas (UCI) no existen resultados históricos de proyectos ya finalizados que permita realizar estimaciones y análisis detallado tanto cualitativo como cuantitativo para lograr una estimación concreta, por lo que la planificación de los proyectos en muchas ocasiones no es realizada con la calidad que se requiere ,y no se puede llegar a una idea exacta del costo de desarrollo y de soporte , de la cantidad de horas trabajadas, del tiempo transcurrido, así como de la distribución del esfuerzo por fase, influyendo esto en la demora de entrega de los mismos, así como en la eficiencia y eficacia del producto. No existen métricas que permitan medir el proceso de desarrollo de software en la universidad, constituyendo así la **situación problémica** de la investigación.

Este trabajo se centra en realizar la propuesta de métricas de software para la estimación en el desarrollo de cualquier software en la universidad, además de analizar cuales son los métodos de estimación más factibles de aplicar en la universidad para un desarrollo eficiente de los diferentes proyectos.

Es por ello que surge esta investigación planteándose como **problema científico**: ¿Cómo se puede establecer una serie de indicadores estables para realizar las estimaciones de futuros proyectos productivos en la Universidad de las Ciencias Informáticas (UCI)?

A modo de solución al problema planteado se hace necesario un estudio profundo y detallado de las métricas existentes y la selección de la que se ajuste al marco definido en la universidad por lo que se define como **objeto de estudio** las métricas a utilizar para estimar en proyectos de software.

Como **campo de acción** se define: las métricas para estimar en el desarrollo de los proyectos de software que se realiza en la universidad.

Para resolver los problemas que se plantearon con anterioridad, se propone como **objetivo general**: Proponer métricas para estimar y controlar el avance del proceso de desarrollo de software en la universidad. Para ello se trazaron los siguientes **objetivos específicos** de la investigación:

- Estudiar el estado actual de las métricas para estimar en los proyectos de software en Cuba y en el mundo.
- Analizar la existencia de métricas para estimar utilizadas en proyectos productivos en la universidad.
- Proponer métricas para estimar y controlar el avance de los proyectos de software en la UCI.
- Validar la propuesta.

Para el desarrollo de la investigación se parte de la **idea a defender** que: Si se proponen métricas para estimar el proceso de desarrollo de los proyectos de software en la universidad, se espera que se eleve la calidad del software que se produce.

Para dar cumplimiento a los objetivos presentados anteriormente se concretan las siguientes **tareas de investigación:**

- Estudio de las métricas para estimar que se utilizan en el desarrollo del software en Cuba y en el mundo.
- Realización de análisis de las métricas para estimar utilizadas en la universidad.
- Procesamiento y evaluación de la información obtenida.
- Realización de la propuesta de las métricas a utilizar.
- Realización de la evaluación técnica de la propuesta.

El presente trabajo consta de introducción, 3 capítulos y conclusiones.

El Capítulo 1: **Fundamentación Teórica** aborda los temas relacionados con las métricas, conceptos, tipos y reflexiones que colaboran al esclarecimiento de la importancia de su aplicación en los proyectos productivos.

El Capítulo 2: **Propuesta de las métricas de software para estimar los proyectos de la universidad:** en este capítulo se realiza un estudio de los proyectos productivos de la universidad, y se analiza la aplicación de las métricas en estos, y a partir de aquí, se definen las métricas propuestas, estableciendo una estrategia para su uso en la universidad.

El Capítulo 3: **Validación de la propuesta:** en este capítulo se realizará el análisis de la evaluación técnica de la propuesta por el métodos de expertos.

Capítulo 1: Fundamentación Teórica

Realizando un análisis detallado del desarrollo y evolución de los proyectos de software en la actualidad se pudiera comenzar destacando el avance de las tecnologías de la información, permitiendo que las tareas realizadas por las personas sean muchos más fáciles, lo que trae consigo enormes beneficios ya que permite de forma eficiente el ahorro de tiempo. Solo que no todo es tan simple, sería bueno preguntarse: ¿Qué tan eficientes son estas tareas? ¿Cómo podrían ser más eficientes? Y ¿Qué tanto se puede confiar en un programa más que en una persona para realizar estas tareas?

Desde hace mucho tiempo han sido muchas las organizaciones, empresas e instituciones que han abordado esta problemática, algunas con mayor rigor que otras, pero todas en aras de analizar el problema en el desarrollo de sistemas de software. Las diferentes investigaciones realizadas han sido encaminadas a la búsqueda de las causas y a la presentación de temas proponiendo reglas, estándares de procesos, prácticas necesarias, métodos de estimación y métricas que aborden el desarrollo, mantenimiento y operación de un producto de software con el objetivo de obtener resultados exitosos.

Con el avance de la tecnología, el desarrollo se ha evidenciado en todos los sectores de la sociedad, convirtiéndose así en el eje principal de diferentes esferas sociales, donde se han obtenido logros significativos que han marcado pautas en la historia y progreso de diferentes naciones dando lugar a la era de la información. Según van evolucionando las tecnologías, el mercado mundial se desborda de productos de software cada vez más eficientes, pudiéndose observar el auge de la competencia actual por obtener productos óptimos y eficientes, lo que trae consigo que exista la necesidad de exigir mayor precisión a la hora de estimar, para planificar y medir cuánto esfuerzo, recursos y tiempo será necesario para construir un producto de software con la calidad requerida. De ahí que sea importante explicar en qué consiste la calidad:

1.1 Calidad

A lo largo del capítulo se ha mencionado más de una vez la palabra calidad sin una previa definición, es válido aclarar que depende en gran medida de la evaluación y criterio del público al que esté dirigido un producto o proceso en cuestión. A lo largo de la historia ha evolucionado dependiendo de las necesidades del contexto histórico y los objetivos a perseguir. Ahora vale preguntarse: ¿Qué es calidad?

Según la Norma ISO 8402, se define Calidad como la “totalidad de las características de un producto o servicio que le confieren su aptitud para satisfacer unas necesidades expresadas o implícitas”[1] por su parte, el Diccionario de la Real Academia Española, conceptúa la Calidad como la "propiedad o conjunto de propiedades inherentes a algo, que permiten juzgar su valor", y es sinónimo de "buena calidad" la "superioridad o excelencia".[2]

Se considera que Calidad, resume las características, propiedades, cualidades y en general atributos propios de un producto, que determinan sobre este la ausencia de defectos y la conformidad de todo el personal que de una forma u otra se vinculan con él, (productores, clientes, usuarios), es además la asociación de varios conceptos que se mencionan a continuación:

Sistema de Calidad: Conjunto de la estructura, responsabilidades, actividades, recursos y procedimientos de la organización de una empresa. ¿Qué sistema se establece para garantizar que lo que ofrece cumple con las especificaciones establecidas previamente por la empresa y el cliente, asegurando una calidad continua a lo largo del tiempo?

Control de calidad*:* El control está dirigido al cumplimiento de requisitos no es más que un conjunto de actividades y técnicas operativas, utilizadas para verificar los requerimientos relativos a la calidad del producto o servicio.

Garantía de calidad: Conjunto de acciones planificadas y sistemáticas necesarias para proporcionar la confianza adecuada de que un producto o servicio satisface los requerimientos dados sobre calidad, o sea tiene como finalidad inspirar confianza en que se cumplirá el requisito pertinente.

Gestión de la calidad: La gestión de la calidad es un conjunto de actividades y medios necesarios para definir e implantar un sistema de la calidad, responsabilizarse de su control, aseguramiento y mejora continua.

1.1.1 ¿Qué es Calidad de Software?

Se define la calidad de software como la ausencia de errores de funcionamiento, la adecuación a las necesidades del usuario, y el alcance de un desempeño apropiado (tiempo, volumen, espacio), además del cumplimiento de los estándares. Los objetivos que la calidad persigue son: La aceptación (utilización real por parte del usuario) y la mantenibilidad (posibilidad y facilidad de corrección, ajuste y modificación durante largo tiempo).[3]

La calidad del software es definida por Pressman como la concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente

documentados y con las características implícitas que se esperan de todo software desarrollado profesionalmente.[3]

Se considera que la calidad no es una simple característica del software que se aprecia en el producto final, sino el resultado de los esfuerzos de todos los miembros del equipo durante todas las etapas del proceso de desarrollo del software (planeación, análisis, diseño, programación, pruebas, mantenimiento). Por lo general se tiene en cuenta la satisfacción del cliente, la entrega oportuna, la importancia de las mediciones y métricas, así como el desarrollo de un proceso bien definido. Uno de los elementos que permite dar garantía acerca de la calidad del software es la aplicación de métricas.

Para lograr la calidad las empresas de software trabajan en pos de certificar sus productos teniendo en cuenta algunos de los modelos de calidad existentes, uno de ellos es el CMMI y a continuación se explica los términos más importantes del mismo.

1.1.1.1 ¿Qué es CMMI?

CMMI (Modelo de Capacidad y Madurez Integrado) es un modelo que constituye un marco de referencia de la capacidad de desarrollo de software de diferentes empresas, siendo una base para evaluar la madurez de las mismas y una guía para llevar a cabo una estrategia de mejora continua. Este modelo surgió a partir de CMM (Modelo de Capacidad y Madurez) como resultado de la integración de varios modelos que definían la madurez en diferentes disciplinas, pero que dada la variedad constituían un problema para las empresas por lo que fue necesario agruparlos en el CMMI.

La capacidad del proceso de software describe el rango de resultados esperados que se obtienen siguiendo un proceso de software, mientras que el desempeño representa los resultados reales obtenidos. La madurez del proceso de software se refiere a cuán explícitamente definido, administrado, medido, controlado y efectivo ha sido un proceso en específico.

CMMI dirige su enfoque a la mejora de procesos en una organización, los estudia y mide la capacidad para construir un software de calidad, atendiendo a una escala de cinco niveles (inicial, repetible, definido, dirigido y optimizado). Para que cada organización pueda enfocar la mejora de sus procesos se especifica en cada nivel de madurez un conjunto de áreas de proceso, que se describen en términos de prácticas, estas son un conjunto de actividades que contribuyen a la implementación eficiente de un área de proceso, si se cumplen todas las prácticas y se satisfacen todas las áreas de proceso de un determinado nivel entonces la empresa podrá certificarse en ese nivel de madurez.[4]

En CMMI como en otros modelos de Calidad se tiene en cuenta las mediciones como datos necesarios para el cálculo de métricas. Estas deben ser recepcionadas por cada desarrollador de software en

cualquier proyecto, o en general por el equipo de trabajo. En el mundo de las tecnologías de la información se han definido procesos que apoyen el trabajo individual y en equipo para el desarrollo de cualquier proyecto de software. Un ejemplo de ello son los procesos definidos por Watts S. Humphrey conocidos como: Proceso Personal de Software y el Proceso de Software del Equipo, de los cuales se hablará a continuación.

1.1.1.2 Proceso Personal de Software (PSP)

El Proceso Personal de Software (PSP) es un proceso de auto mejoramiento diseñado para ayudar a controlar, administrar y mejorar la forma en que se trabaja individualmente. Está estructurado por formularios, guías y procedimientos para desarrollar software. Si es usado apropiadamente, brinda los datos históricos necesarios para trabajar mejor y lograr que los elementos rutinarios del trabajo sean más predecibles y eficientes.

Usando PSP, se pueden construir programas de más de 10 000 líneas de código (LOC), sin embargo, hay dos problemas típicos en los grandes programas. Primero, mientras se crece en tamaño, también lo hace el tiempo y el esfuerzo requerido. Esto puede ser una situación particular si solo existe un ingeniero en el proyecto.

Segundo, la mayoría de los ingenieros tienen problemas en la visualización de todas las facetas importantes de un programa, incluso cuando su tamaño es moderado. Existen muchos detalles e interrelaciones que deben tenerse en cuenta, muchas dependencias lógicas, interacciones en el tiempo o condiciones excepcionales. Una de las formas más poderosas de resolver estos problemas es el Proceso de Software en Equipo (TSP).

1.1.1.3 Proceso de Software del Equipo (TSP)

Una definición acertada de un equipo es la dada por Dyer, donde un equipo consiste en al menos dos personas que trabajan para lograr una meta, objetivo, misión común, donde cada persona tiene asignado un rol específico o funciones específicas que desarrollar, y donde el completamiento de la misión requiere alguna forma de dependencia entre los miembros del equipo. El TSP es un proceso que al igual que el PSP, está basado en el modelo CMMI, TSP está diseñado para ayudar a controlar, administrar y mejorar la forma en que trabaja un equipo de software. Al igual que PSP, está estructurado por formularios, guías y procedimientos para desarrollar software.

1.1.1.4 Relación entre PSP, TSP y CMM

- CMM/CMMI, TSP y PSP pueden usarse de forma 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.

Dentro de la calidad se encuentran las métricas del software las cuales permiten medir el software y obtener mayores resultados de este. A continuación se abordarán los diferentes temas relacionados con las métricas de software:

1.2 Métricas de software

Las organizaciones no habían comprendido la necesidad de medir para obtener mejores indicadores hasta la aparición del mercado competitivo. Existen problemas en organizaciones a la hora de ponerse de acuerdo en aspectos como qué se debe medir y cómo hacerlo.

Dentro del contexto de Ingeniería de Software, una medida “proporciona una indicación cuantitativa de extensión, cantidad, dimensiones, capacidad y tamaño de algunos atributos de un proceso o producto”.[5]

Por su parte la medición es “el proceso por el cual se asignan números o símbolos a atributos de entidades del mundo real de tal manera que describa dichos atributos de una forma significativa y de acuerdo a unas reglas claramente definidas”[6], también es definida como el acto de determinar una medida.

El primer paso de la medición es identificar los atributos o entidades a medir. Estos pueden ser de tres tipos:

Procesos: Atributos de actividades relacionadas con el software.

Productos: Componentes, entregas o documentos resultantes de una actividad de proceso.

Recursos: Entidades requeridas por una actividad de proceso.

Las métricas se pueden definir como:

La IEEE “Glosario Estándar de Términos Ingenieros de Software” define métrica como “una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado”.[7] “Las métricas son un buen medio para entender, monitorizar, controlar, predecir y probar el desarrollo software y los proyectos de mantenimiento”.[8]

Una métrica según [9] es una serie de medidas o pasos que nos ayudan a definir con mayor exactitud el desarrollo y calidad de un producto. Estas pueden ser cualitativas, es decir, nos expresan en palabras una medida de como ha evolucionando el producto de software, y también cuantitativas, esta vez nos dan una medida del software pero con cifras, y sirven para indicar la calidad del producto, evaluar el trabajo y desempeño de una persona, el uso de nuevas herramientas, obtener estimaciones más exactas de proyectos futuros más acertados.

Para que sea útil en el contexto del mundo real, una métrica del software debe ser objetiva, simple y calculable, consistente en el empleo de unidades y tamaños, persuasiva, además debería ser independiente del lenguaje de programación y proporcionar una retroalimentación eficaz para el desarrollador de software.[5]Las métricas de software proveen la información necesaria para la toma de decisiones técnicas. En la figura1 se ilustra una extensión de esta definición para incluir los servicios relacionados al software como la respuesta a los resultados del cliente.

Las métricas de software las han definido como: “La aplicación continua de mediciones basadas en técnicas para el proceso de desarrollo del software y sus productos para suministrar información relevante a tiempo, así el planificador junto con el empleo de estás técnicas mejorará el proceso y sus productos”.

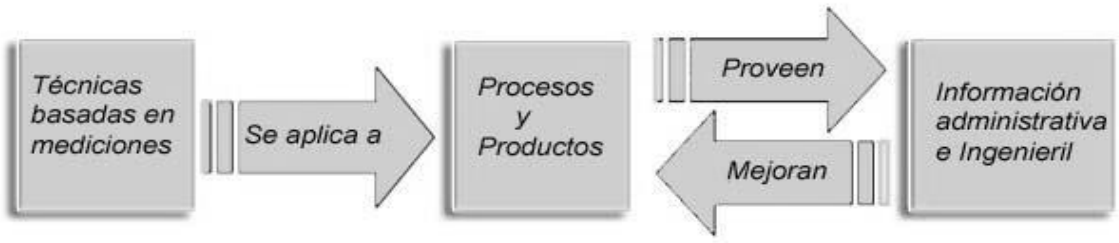


Figura 1.Métricas de Software

Las métricas son la maduración de una disciplina, van a ayudar a la evaluación de los modelos de análisis y de diseño, proporcionarán una indicación de la complejidad de diseños procedimentales y de código fuente, y 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[5]:

- 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.
- Retroalimentación: Recomendaciones obtenidas de la interpretación de métricas técnicas trasmitidas al equipo de software.

1.2.1 Diferentes enfoques de métricas

Muchas son las métricas o medidas propuestas con el objetivo que proporcionen una medida completa de la complejidad del software. Muchos son los investigadores que han intentado desarrollar una métrica, pero no todas son los completamente eficientes como para que proporcionen suficiente soporte práctico para su desarrollo. Entre las diferentes dificultades que se pueden encontrar se tiene que algunas demandan mediciones que son demasiado complejas, otras son tan complicadas 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 obligaría 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 retroalimentación 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.

1.3 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 Directas o Indirectas, Primarias o Secundarias, Internas o Externas, Públicas o Privadas, Simples o Complejas, de Proceso, de Producto o de Proyecto, Primitivas o Calculadas.

Pressman clasifica el campo de las métricas en seis 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 se va a diseñar.

- Métricas orientadas al tamaño. Es para saber en qué tiempo se va a terminar el software y cuántas personas se van 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 persona que 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 se van a hacer del personal que hará el sistema.

Estas clasificaciones de métricas fortalecen la idea, de que más de una métrica puede ser deseable para valorar la complejidad y la calidad del software, teniendo en cuenta que para ello es necesario medir los atributos del software.

1.3.1 Métricas privadas y públicas

En el proceso pueden incluirse métricas privadas y públicas. Las métricas privadas como su nombre lo indica solo le conciernen a un individuo en particular, cada trabajador debe ser capaz de controlar sus datos individuales durante el proceso de desarrollo, esto le ayudará a corregir sus errores. Claro está que las métricas no deben emplearse para evaluar el desempeño de los programadores sino como un indicador individual que le permita al programador mejorar su trabajo, lo que evidentemente influirá en el resultado final del proyecto. Entre las métricas privadas se incluyen: índices de defectos (individuales), índices de defectos (por módulo) y errores encontrados durante el desarrollo.

Las métricas pueden ser públicas para todos los miembros de un equipo del proyecto de software y sin embargo ser consideradas como privadas para ese equipo, por otra parte las métricas públicas para la organización proporcionan beneficios significativos para mejorar el nivel global de madurez del proceso, pues generalmente asimilan información acerca del esfuerzo, tiempo y datos afines que se recopilan y se evalúan en un intento de detectar indicadores que puedan mejorar el rendimiento del proceso organizativo.

1.3.2 Métricas directas e indirectas

Una métrica directa no es más que una métrica de la cual se pueden realizar mediciones sin depender

de ninguna otra medición y cuya forma de medir es un método de estimación. Una métrica directa puede ser utilizada en funciones de cálculo.

Ejemplo de métricas directas

- LCF (línea de código fuente escritas)
- HPD (horas-programador escritas)

Por su parte una métrica indirecta es una métrica cuya función de cálculo, es decir, las mediciones de dicha métrica utilizan las medidas obtenidas en mediciones de otras métricas indirectas o directas.

Ejemplo de métricas indirectas

- HTP (hora-programador totales)
- CTP (coste actual del proyecto, en unidades monetarias)
- CLCF (coste por línea de código fuente)

1.3.3 Métricas internas y externas

La métrica interna puede ser aplicada a un producto de software no ejecutable durante el diseño y la codificación. El propósito primario de esta métrica interna es asegurar que se logre la calidad externa y la calidad de uso requerida. La métrica interna proporciona a los usuarios, evaluadores, verificadores y desarrolladores el beneficio de que puedan evaluar la calidad del producto de software y lo referido a problemas de calidad antes que el producto de software sea puesto en ejecución. Las métricas internas miden atributos internos o indican los atributos externos, a través del análisis de las propiedades estáticas de productos intermedios o entregables del software. Las medidas de las métricas internas usan números o frecuencias de elementos de composición de software, los cuales aparecen, por ejemplo, en las sentencias de código de fuente, control de gráficos, flujo de datos y estados de representación de procesos.

Las métricas externas usan medidas de un producto de software, derivadas del comportamiento del mismo, a través de la prueba, operación y observación del software. Antes de adquirir o usar un producto de software, éste debe ser evaluado usando las métricas basadas en los objetivos del área usuaria de la institución relacionados al uso, explotación y dirección del producto, considerando la organización y el ambiente técnico. La métrica externa proporciona a los usuarios, evaluadores, verificadores y desarrolladores, el beneficio de que puedan evaluar la calidad del producto de software durante las pruebas o el funcionamiento.

Relación entre métricas internas y externas

Cuando los requisitos de calidad del producto de software son definidos, se listan las características de calidad del producto de software que contribuyen a dichos requisitos. Entonces, las métricas externas apropiadas y los rangos aceptables son especificados para cuantificar el criterio de calidad que valida que el software satisface las necesidades del usuario.

Algunas métricas internas son especificadas para cuantificar los atributos de calidad interna, así ellos pueden usarse para verificar que el software intermedio reúne las especificaciones de calidad interna durante el desarrollo.

Se recomienda que las métricas internas que se usen tengan en lo posible una fuerte relación con la métrica externa diseñada, para que ellas puedan ser usadas para predecir los valores de las métricas externas. Sin embargo, es generalmente difícil diseñar un modelo teórico riguroso que proporcione una relación fuerte entre la métrica interna y la externa.

1.3.4 Métricas del proceso

En las métricas del proceso se recopilan de todos los proyectos durante un largo período de tiempo. Su intento es proporcionar indicadores que lleven a mejoras de los procesos de software a largo plazo.[5] Un indicador es una métrica o una combinación de métricas que proporcionan una visión profunda del proceso del software, del proyecto o del producto en sí.

La medición del proceso implica las mediciones de las actividades relacionadas con el software siendo algunos de sus atributos típicos el esfuerzo, el coste y los defectos encontrados. Las métricas permiten tener una visión profunda del proceso de software que ayudará a tomar decisiones más fundamentadas, ayudan a analizar el trabajo desarrollado, conocer si se ha mejorado o no con respecto a proyectos anteriores, ayudan a detectar áreas con problemas para poder remediarlos a tiempo y a realizar mejores estimaciones.

Para mejorar un proceso se deben medir los atributos del mismo, desarrollar métricas de acuerdo a estos atributos y utilizarlas para proporcionar indicadores que conduzcan la mejora del proceso. Los errores detectados antes de la entrega del software, la productividad, los recursos y el tiempo consumido, y el ajuste con la planificación son algunos de los resultados que pueden medirse en el proceso, así como las tareas específicas de la Ingeniería del Software.

Las métricas del proceso se caracterizan por:

- El control y ejecución del proyecto.

- Medición de tiempos del análisis, diseño e implementación.
- Medición de las pruebas (errores, cubrimiento, resultado en número de defectos y número de éxito).
- Medición de la transformación o evolución del producto.

1.3.5 Métricas del proyecto

Dado que el proyecto engloba todos los recursos, actividades y artefactos, que se organizan para lograr un producto de software es de vital importancia definir algunas mediciones que ayuden al mejoramiento del mismo. A nivel de proyecto se minimiza la planificación de desarrollo haciendo los ajustes necesarios para evitar retrasos o riesgos potenciales, minimizar los defectos, y por tanto la cantidad de trabajo que ha de rehacerse, lo que ocasiona una reducción del coste global del proyecto, además puede evaluarse la calidad de los productos en el momento actual y cuando sea necesario.

La primera aplicación de métricas de proyectos en la mayoría de los proyectos de software ocurre durante la estimación. Las métricas recopiladas de proyectos anteriores se utilizan como una base desde la que se realizan las estimaciones del esfuerzo y del tiempo para el actual trabajo del software. A medida que avanza un proyecto, las medidas del esfuerzo y del tiempo consumido se comparan con las estimaciones originales (y la planificación de proyectos). El gestor de proyectos utiliza estos datos para supervisar y controlar el avance. A medida que comienza el trabajo técnico, otras métricas de proyectos comienzan a tener significado. Se miden los índices de producción representados mediante páginas de documentación, las horas de revisión, los puntos de función y las líneas fuentes entregadas, en el proyecto se sigue la pista de los errores detectados durante todas las tareas de Ingeniería del Software. Cuando va evolucionando el software desde la especificación del diseño, se recopilan las métricas técnicas para evaluar la calidad del mismo y para proporcionar indicadores que influirán en el enfoque tomado para la generación y prueba del código.[5] Finalmente los indicadores de proyecto permiten:

- Evaluar el estado del proyecto en curso.
- Seguir la pista de los riesgos potenciales.
- Detectar las áreas de problemas antes de que se conviertan en “críticas”.
- Ajustar el flujo y las tareas del trabajo.

- Evaluar la habilidad del equipo del proyecto y controlar la calidad de los productos de trabajo del software.

RUP plantea que las principales métricas a tener en cuenta son [10]:

- Modularidad: Promedio de daños debido a cambios perfectivos o correctivos en la implementación.
- Adaptabilidad: Promedio de esfuerzo debido a cambios perfectivos o correctivos en la implementación.
- Madurez: Tiempo de prueba activo / número de cambios correctivos.
- Mantenimiento: Mantenimiento productivo / desarrollo productivo.
- Progreso del proyecto: Debe reportarse basándose en el plan del proyecto desde la perspectiva del valor devengado.

1.3.6 Métricas del producto

Las métricas del producto se centran en las características del software y no en cómo fue producido. Un producto no es solo el software o sistema funcionando sino también los artefactos, documentos, modelos, módulos, o componentes que lo conforman, por tanto, las métricas del producto deben hacerse sobre la base de medir cada uno de estos.

En los artefactos del producto se mide, entre otras cosas, el tamaño, la calidad (teniendo en cuenta los defectos y la complejidad), la totalidad, rastreabilidad, volatilidad, esfuerzo.

1.3.6.1 Relación entre el Proceso y el Producto

Un proceso puede ser valorado indirectamente midiendo y evaluando el producto, y un producto puede evaluarse indirectamente midiendo la ejecución de las tareas de los usuarios específicos que verifican que el objetivo planteado se haya logrado con eficacia, productividad, seguridad y satisfacción. En las primeras etapas de desarrollo solo pueden medirse el proceso y los recursos, cuando se ha obtenido productos intermedios estos pueden ser medidos utilizando métricas internas que a su vez pueden usarse para predecir valores de las métricas externas. En la Figura 2 se muestra la Calidad en el ciclo de vida del software.

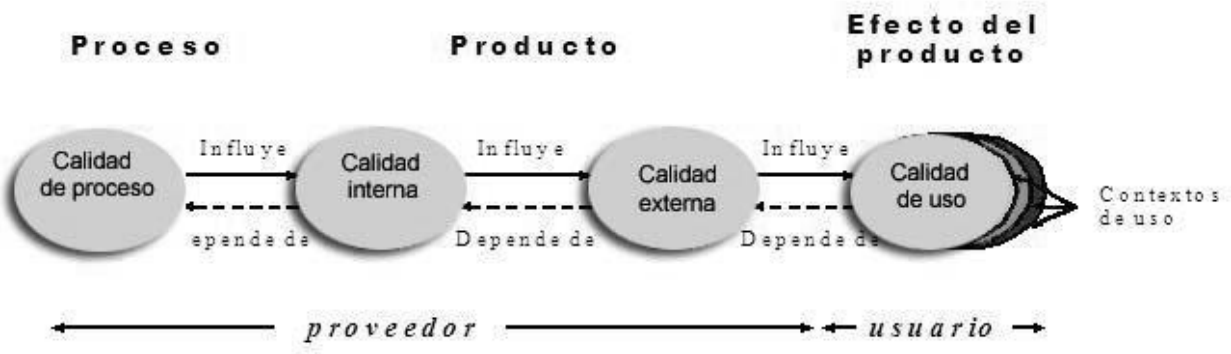


Figura 2.Calidad en el ciclo de vida del software.

1.3.6.2 Métricas y Calidad

El principal objetivo de los ingenieros del software es producir un sistema, aplicación o producto de alta calidad, para lo cual emplean métodos y herramientas efectivas dentro del contexto de un proceso maduro de desarrollo del software y además deben desarrollar mediciones que den como resultado sistemas de alta calidad. Para obtener esta evaluación, el ingeniero debe utilizar medidas técnicas, que evalúan la calidad con objetividad, no con subjetividad.

1.4 Estimación

La estimación es un tema que está estrechamente relacionado con las métricas y del cual no habíamos abordado anteriormente, a continuación se hablará de todo lo que respecta en cuanto a la estimación.

La estimación de la duración de las actividades que conforman el desarrollo de software es un tema que concierne a la gestión y control de proyectos. La estimación es una pequeña planeación sobre que es lo que va a ser en un proyecto. La primera tarea en la gestión de proyectos es la estimación.[11]

La estimación está definida como “el proceso que proporciona un valor a un conjunto de variables para la realización de un trabajo, dentro de un rango aceptable de tolerancia”.[11]

Otras de las definiciones encontradas de lo que es la estimación; es la siguiente: “la estimación es la predicción del personal, del esfuerzo, de los costos y del tiempo que se requerirán para realizar todas las actividades y construir todos los productos asociados con el proyecto”.[12]

Se comienza la estimación haciendo uso de pocas variables en un nivel alto de abstracción, permitiéndose obtener valores aproximados de costo, tiempo y esfuerzo como para estudiar la

viabilidad del proyecto. Una vez comenzado el proyecto y obtenidos estos valores se pueden efectuar comparaciones, detectar desvíos en el plan y realizar los ajustes correspondientes en lo que respecta a una buena planificación.

El desarrollo del software requiere de la estimación para controlar y administrar los recursos que se necesitan utilizar antes y durante el proyecto. No se puede considerar la estimación como una ciencia exacta ya que existen numerosas variables humanas, técnicas, del entorno y políticas, que intervienen en su proceso y que pueden afectar los resultados finales. Sin embargo, cuando es llevada a cabo en forma sistemática, se pueden lograr resultados con un grado aceptable y convertirla en un instrumento útil para la toma de decisiones.

Cualquier cronograma de tareas pendientes implica estimaciones. Cualquier intención de calcular tiempos, recursos o costos al futuro, implica estimaciones. Sin estimaciones no podría haber gestión de proyectos porque el proyecto es, como su nombre lo indica, una proyección.

La estimación, como actividad antecesora y constante de lo que luego será la planificación, proporciona valores aproximados de costos, tiempos y esfuerzos que se necesitarán para el desarrollo del producto a construir. Esa aproximación, requiere experiencia, acceder a una buena información histórica y el coraje de confiar en predicciones cuantitativas cuando todo lo que existe son datos cualitativos;[5] sin embargo, contar con dicha información en etapas tempranas de desarrollo, permite tomar decisiones importantes antes llevarlo a cabo, es decir que de cierta forma nos estamos atreviendo a predecir el futuro

Por otra parte una definición no técnica de estimación dice que es un conjunto aproximado de valores para algo que ha de ser hecho.

La correcta estimación temprana de un proyecto de software es una tarea difícil o casi imposible. Sin embargo la mayor parte de los sistemas que son desarrollados por terceros, requieren fijar un precio antes de contratarse el desarrollo. La estimación es necesaria para la definición de ese precio.

Un proyecto de software requiere obtener estimaciones de esfuerzo humano, de la duración cronológica del esfuerzo requerido, del proyecto y del costo. Pero en muchos de los casos las estimaciones se hacen valiéndose de la experiencia pasada como única guía. Si un proyecto es bastante similar en tamaño y funcionamiento y si dicho proyecto es bastante similar en tamaño y funciona como un proyecto pasado es probable que el nuevo proyecto requiera aproximadamente la misma cantidad de esfuerzo, que dure aproximadamente lo mismo que el trabajo anterior. Pero, ¿Qué

pasa si el proyecto es totalmente distinto?, entonces puede que la experiencia obtenida no sea lo suficiente.

Se han desarrollado varias técnicas de estimación para el desarrollo de software, aunque cada una tiene sus puntos fuertes y sus puntos débiles, todas tienen en común los siguientes atributos.

1. Se han de establecer de antemano el ámbito del proyecto.

2. Como bases para la realización de estimaciones se usan métricas del software de proyectos pasados.

3. El proyecto se desglosa en partes más pequeñas que se estiman individualmente.

La estimación es siempre difícil de realizar por diversas razones, algunas de ellas son [13]:

- No existe un modelo de estimación universal.
- Son muchas las personas implicadas en el proyecto, desde la alta dirección de la empresa a los ejecutivos del proyecto, que precisan de las estimaciones.
- La utilidad de una estimación varía con la etapa de desarrollo en que se encuentra el proyecto.
- Las estimaciones precisas son difíciles de formular, sobre todo al inicio del proyecto.
- Las estimaciones suelen hacerse superficialmente, sin tener en cuenta el esfuerzo necesario para hacer el trabajo.
- La rapidez del cambio de las metodologías y las tecnologías no permiten la estabilización del proceso de estimación.
- Los estimadores pueden no tener experiencias sobre aquello que pretenden estimar.
- El estimador suele hacer la estimación en función del tiempo que a él le llevaría en realizar el trabajo, sin tener en cuenta la experiencia y formación de la persona que realmente lo realiza. Este tiende a reducir en alguna medida sus estimaciones para hacer más aceptable su oferta.

1.4.1 Planeación del proyecto

La planeación efectiva de un proyecto de software depende de la planeación detallada de su avance, anticipando problemas que puedan surgir y preparando con anticipación soluciones tentativas a ellos. Se supondrá que el administrador del proyecto es responsable de la planeación desde la definición de requisitos hasta la entrega del sistema terminado. No se analizará la planeación que implica a la

estimación de la necesidad de un sistema de software y la habilidad de producir tal sistema, la asignación de prioridad al proceso de su producción.

Entre los aspectos a tener en cuenta en la planeación se puede encontrar:

- Panorama: Hace una descripción general del proyecto en detalle de la organización del plan y resume el resto del documento.
- Plan de fases: Se analiza el ciclo de desarrollo del proyecto como es: análisis de requisitos, fase de diseño de alto nivel, fase de diseño de bajo nivel, etc. Asociada con cada fase debe de haber una fecha que especifique cuando se debe terminar estas fases y una indicación de como se pueden solapar las distintas fases del proyecto.
- Plan de organización*:* Se definen las responsabilidades específicas de los grupos que intervienen en el proyecto.
- Plan de pruebas: Se hace un esbozo general de las pruebas, herramientas, procedimientos y responsabilidades para realizar las pruebas del sistema.
- Plan de control de modificaciones: Se establece un mecanismo para aplicar las modificaciones que se requieran a medida que se desarrolle el sistema.
- Plan de documentación: Su función es definir y controlar la documentación asociada con el proyecto.
- Plan de capacitación: Se describe la preparación de los programadores que participan en el proyecto y las instrucciones a los usuarios para la utilización del sistema que se les entregue.
- Plan de revisión e informes: Se analiza como se informa del estado del proyecto y se definen las revisiones formales asociadas con el avance de proyecto.
- Plan de instalación y operación*:* Se describe el procedimiento para instalar el sistema en la localidad del usuario.
- Plan de recursos y entregas*:* Se resume los detalles críticos del proyecto como fechas programadas, marcas de logros y todos los artículos que deben entrar bajo contrato.
- Índice: Se muestra en donde encontrar las cosas dentro del plan.
- Plan de mantenimiento*:* Se establece un bosquejo de los posibles tipos de mantenimiento que se tienen que dar para futuras versiones del sistema.

1.4.2 Errores clásicos en un proyecto de software

- Mal análisis en los requerimientos.
- Una mala planeación.
- No tener una negociación (documento, contrato) con el cliente.
- No hacer un análisis costo beneficio.
- Desconocer el ambiente de trabajo de los usuarios.
- Desconocer los usuarios que trabajan con el sistema.
- Mala elección de recursos (hardware, software, humanos).

1.4.3 Estimación de Costos

La estimación y la realización del cronograma del proyecto se llevan a cabo de forma conjunta. Sin embargo, en las primeras etapas del proyecto se requieren algunas estimaciones de costos, antes que se tenga el cronograma detallado. Estas estimaciones son necesarias para establecer un presupuesto para el proyecto o para asignar un precio para el software de un cliente.

Una vez que el proyecto se comienza a ejecutar, las estimaciones se actualizan de forma regular. Esto ayuda al proceso de planeación y permite la utilización efectiva de los recursos. Si el gasto real es significativamente más grande que las estimaciones, entonces el administrador del proyecto debe tomar algunas acciones. Éstas pueden ser: solicitar recursos adicionales para el proyecto o modificar el trabajo a realizar.

Existen tres parámetros involucrados en el cálculo del costo total de un proyecto de desarrollo de software:

- Los costos de hardware y software, incluyendo el mantenimiento.
- Los costos de viajes y capacitación.
- Los costos de esfuerzo (los costos de pago a los ingenieros de software.

Para muchos proyectos, el costo dominante es el costo del esfuerzo. Las computadoras que son suficientemente poderosas como para desarrollar el software son relativamente baratas. Aunque los costos de viajes pueden ser importantes si un proyecto se desarrolla en sitios distintos, son relativamente bajos para muchos proyectos. Además, el uso de correo electrónico, los chat, fax y teleconferencias reduce los viajes requeridos.

1.5 Métodos de Estimación

Los métodos de estimación deben basarse en un parámetro o elemento de estimación que tenga las siguientes características:

- Objetivo.
- Fácilmente identificable.
- Apto para ser valorado numéricamente.
- Válido.
- Apto para ser refinado a medida que se obtiene mayor información.

En el marco de trabajo durante el proceso de desarrollo de software se clasifican los siguientes métodos de estimación:

- Empíricos.
- Analógicos.
- Teóricos.
- Heurísticos.
- Las estimaciones global y detallada.
- Juicio del experto.
- Método Puntos de Casos de Uso.
- Modelo COCOMO.
- Puntos de Función.
- COCOMO II y los Puntos de Función.

1.5.1 Empíricos

Cualquier estimación se debe basar en un modelo empírico, o sea, algo subjetivo producto de la experiencia, que relacione un atributo de interés con otros atributos mensurables. Este modelo empírico es el punto de partida para cada método de estimación.

1.5.2 Analógicos

Este método hace la estimación de un proyecto nuevo por analogía con las estimaciones de proyectos anteriores comparables y que estén terminados. En la analogía pueden variar los siguientes factores como el tamaño, complejidad y usuarios.

Utiliza medidas de los atributos del modelo empírico a fin de caracterizar el caso actual, para el que se realiza la estimación. Las medidas conocidas para el caso actual son usadas para buscar un conjunto de datos que identifiquen casos análogos. La predicción se hace intercalando desde uno o varios casos análogos al caso actual.

Las ventajas que proporciona este método es un menor costo en tiempo y recursos que el método del juicio del experto. Como desventajas se puede destacar que las estimaciones de proyectos anteriores no siempre se ajustan a nuevos proyectos, ya que muchos de los factores de estas estimaciones no siempre se mantienen.[14]

1.5.3 Teóricos

Los métodos de estimación teóricos proponen un modelo numérico basado en el modelo empírico. Los modelos teóricos deben ser validados empíricamente, por comparación con los datos actuales de las medidas.

1.5.4 Heurísticas

Los métodos heurísticos suelen usarse como extensiones de otros métodos. Las heurísticas son reglas empíricas, desarrolladas mediante experiencia, que obtienen conocimiento acerca de relaciones entre atributos del modelo empírico. Las heurísticas se pueden utilizar para ajustar estimaciones realizadas con otros métodos.

1.5.5 Las estimaciones global y detallada

La estimación global, conocida también como descendente, se hace teniendo en cuenta las funcionalidades del producto, pasándose posteriormente al detalle.

La estimación detallada o ascendente empieza por la estimación de los esfuerzos individuales, los cuales se suman para obtener el esfuerzo del proyecto.[14]

1.5.6 Juicio del experto

Las opiniones de los expertos se basa principalmente en juicios emitidos por uno o varios expertos avalados por su experiencia en entornos similares y apoyados, en algunos casos, en datos objetivos

obtenidos de proyectos anteriores y almacenados,[14] aunque no se incluyen dentro del marco de trabajo para seleccionar métodos de estimación, ya que estos métodos no se pueden caracterizar fácilmente.

1.5.7 Método Puntos de Casos de Uso

Este método estima el esfuerzo de desarrollo de un producto de software a partir de los Casos de Uso y algunos factores de complejidad técnica y ambiente que influyen en el desarrollo. Fue propuesto originalmente por Gustav Karner y posteriormente refinado por muchos otros autores.

Este método exige la existencia de un modelo de casos de uso, por lo que se deberá comenzar a aplicar, una vez que se tenga algún entendimiento del dominio del problema o cuando se estén realizando las labores de arquitectura y dimensionamiento del tamaño del sistema.[15]

El método utiliza los actores y casos de uso identificados para calcular el esfuerzo que costará desarrollarlos. A los casos de uso se les asigna una complejidad basada en transacciones, que son pares de pasos acción-usuario->respuesta-sistema de los escenarios de los casos de uso. A los actores se les asigna una complejidad basada en el tipo de actor, es decir, si son interfaces con usuarios o si son interfaces con otros sistemas (API o Protocolo). También se utilizan factores de entorno y de complejidad técnica para afinar el resultado.[16]

1.5.8 Modelo COCOMO

COCOMO fue propuesto y desarrollado por Barry Boehm en 1981, es uno de los modelos de estimación de costo mejor documentado, estudiado y utilizado en la industria de software. El modelo permite, basándose en un grupo de ecuaciones no lineales obtenidas mediante técnicas de regresión a través de un histórico de proyectos ya realizados;[17] estimar el esfuerzo, costo y tiempo que se requiere en un proyecto de software a partir de una medida del tamaño del mismo, expresada en el número de líneas de código que se estimen generar para la creación del producto software. El modelo original ha evolucionado a un modelo más completo llamado COCOMO II.

1.5.9 Puntos de Función

El método de puntos de función fue creado por Allan Albretch y se basa principalmente en la identificación de los componentes del sistema informático en términos de transacciones y grupos de datos lógicos que son relevantes para el usuario en su negocio. A cada uno de estos componentes les asigna un número de puntos por función basándose en el tipo de componente y su complejidad; y la

sumatoria de esto, da los puntos de función sin ajustar. El ajuste es un paso final basándose en las características generales de todo el sistema informático que se está contando.

Los objetivos de calcular Puntos de Función son:

- Medir lo que el usuario pide y lo que el usuario recibe.
- Medir atributos independientemente de la tecnología utilizada en la implantación del sistema.
- Proporcionar una métrica de tamaño que de soporte al análisis de la calidad y la productividad.
- Proporcionar un medio para la estimación del software.
- Proporcionar un factor de normalización para la comparación de distintos software.

1.5.10 COCOMO II y los Puntos de Función

Debido a la complejidad de los proyectos de software, el modelo original COCOMO, fue modificado, denominándose al modelo actual COCOMO II. El nuevo modelo permite determinar el esfuerzo y tiempo de un proyecto de software a partir de los puntos de función sin ajustar, lo cual supone una gran ventaja, dado que en la mayoría de los casos es difícil determinar el número de líneas de código de que constará un nuevo desarrollo, en especial cuando se tiene poca o ninguna experiencia previa en proyectos de software. Esto hace que ambos modelos, Puntos de Función y COCOMO sean perfectamente compatibles y complementarios. Su triunfo depende ampliamente de la adaptación del modelo a las necesidades de la organización, usando datos históricos; los cuales no siempre están disponibles.

1.5.11 Puntos de Característica

Debido a que el análisis por Puntos de Función fue diseñado para software de negocios y no es fácil de generalizar a aplicaciones científicas, de tiempo real y otras, Caper Jones propuso ampliaciones a este método, generando una métrica que denominó Puntos de Característica. Ésta da cabida a aplicaciones cuya complejidad algorítmica es alta.

Este método considera los mismos elementos que considera Albrecht en su análisis por puntos de función, sólo que añade la variable "número de algoritmos" y elimina los niveles de complejidad, así, cada cuenta es pesada por un valor único para ese componente (es decir, se le asigna complejidad media).

1.6 Situación en el mundo, en Cuba y en la UCI con respecto a las métricas del software.

A nivel mundial la Industria de Software ha potenciado un gran desarrollo, siendo una de las prioridades lograr la calidad en sus productos y servicios, para lo cual centran sus esfuerzos en la mejora de los procesos de desarrollo teniendo en cuenta normas, estándares y modelos de calidad.

En los últimos años la Industria de software en Cuba ha tenido un desarrollo vertiginoso, y es preocupación de todos los especialistas el tema de la calidad, específicamente la utilización de métricas, pues en su gran mayoría no se tienen recopilados datos históricos de trabajos desarrollados que ayuden a identificar los posibles problemas, ineficiencias u oportunidades de mejora. Uno de los factores que influyen en la calidad del software es la utilización de mediciones que permitan analizar, evaluar, predecir y mejorar los procesos y productos, proporcionando información valiosa para la toma de decisiones y la evaluación del estado en que se encuentran los objetivos fijados El dominio de las métricas del software se divide en métricas de proceso, proyecto y producto.

La Universidad de las Ciencias Informática (UCI) no está aislada del problema de las mediciones. A lo largo de la investigación se han identificado problemas existentes en la Universidad de las Ciencias Informáticas (UCI) los que afectan la eficiencia, la calidad, el tiempo de desarrollo de un producto, la productividad y la efectividad del equipo de desarrollo. Hoy la mayoría de los desarrolladores de software en la UCI: líderes de proyectos, programadores, diseñadores, entre otros roles, no planean ni registran su trabajo y no miden los productos. El desconocimiento de las métricas es un factor que pesa en la evaluación del software.

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.

En la UCI existe un alto porcentaje de proyectos informáticos que ofrecen soluciones artesanales y a medida. En los proyectos no hay definición de roles y responsabilidades que respondan a sus necesidades, afectándose la eficiencia, la calidad, y el tiempo de desarrollo de un producto. Esto empeorará con el aumento de la fuerza de trabajo y de la demanda del cliente. La planificación del trabajo tanto personal como a nivel de equipo no es la mejor, no se siguen estándares establecidos en la Ingeniería de Software, afectándose la efectividad del equipo de desarrollo. El insuficiente dominio

de las herramientas de trabajo provoca que la gestión de las dudas sea un tema crítico e imprescindible, creando dependencia de los líderes.

No se tiene una metodología de estimación y gestión del tiempo de entrega y el costo de un trabajo determinado basado en el conocimiento real y en la capacidad productiva. Logrando así tener un ambiente que nos permita la entrega en tiempo y con calidad. Así como una gestión de cambio.

Este capitulo de manera general se realizó un reconocimiento general sobre el contenido que engloba las métricas de software desde lo conceptualizado y aplicado a nivel mundial hasta la UCI con la problemática actual que como universidad nueva comienza dando los primeros pasos en este tema, por lo que dado el problema en cuestión se realizarán propuestas de métricas con vistas a perfeccionar las estimaciones y planificaciones futuras.

Capítulo 2: Solución Propuesta

Se comienza el capítulo analizando los factores de riesgo que influyen en la elaboración de un buen método de estimación. Se da una explicación del método de estimación elaborado en la UCI. Se analizan además las métricas existentes de manera general, seleccionando las que más se ajusten a las necesidades de información de los proyectos de la UCI una vez estudiada la encuesta realizada, lo que facilitará la obtención de la propuesta de las métricas de software a utilizar en una primera versión en la universidad.

2.1 Análisis de los métodos de estimación existentes

Para dar comienzo al tema sería bueno comenzar desde el análisis de los métodos existentes, los cuales engloban las métricas para realizar las estimaciones. Entonces comenzando desde el nivel más genérico que son los métodos, la primera pregunta a realizarse sería**:** ¿Cómo elaborar un buen método de estimación? Para elaborar un buen método de estimación, es necesario hacer un análisis detallado de algunos factores de riesgo que inciden en las decisiones de las estimaciones antes y después del proceso de desarrollo del software, y controlarlos para de esa forma planear un futuro con más exactitud y que estos factores incidan con mayor precisión sobre las métricas que encierran.

A continuación se brindan algunas características que se desea que deba tener un método de estimación para apoyar la etapa de planificación de los proyectos de desarrollo de software.

- Poder adaptarse a la productividad de la organización.
- Considerar la comunicación entre personas.
- Incorporar guías útiles para estimar aquellos parámetros que son subjetivos o no, que se deducen en forma explícita a partir del modelo.
- Usable.
- Constar de etapas simples de entender y definidas en forma precisa.
- Objetivo.
- Proveer medios para adaptarse a cambios en el ambiente de desarrollo.

2.1.1 Factores de riesgo que inciden en la estimación de un buen método.

El riesgo es la posibilidad de que existan consecuencias indeseables o inconvenientes, de un acontecimiento, cuya aparición no se puede determinar con prioridad. Algunos de los riesgos que influyen en las estimaciones son:

- Evaluación de la organización: Relacionado con la definición de procesos y otras características de la organización.
- Definición del proceso: Grado de definición del proceso de software y su seguimiento por la organización de software.
- Complejidad del sistema a construir: Depende del número de elementos que interactúan entre sí y no sólo de su cantidad sino también de su calidad.
- Tecnología a construir: Asociados con la complejidad del sistema a construir y la tecnología de punta que contiene el sistema.
- Entorno de desarrollo: Disponibilidad y calidad de las herramientas que se van a emplear en la construcción del producto.
- Experiencia del equipo de desarrollo: Experiencia técnica y de proyectos de los desarrolladores que van a realizar el trabajo.

2.1.1.1 Riesgos del software.

Aunque ha habido amplios debates sobre la definición adecuada para riesgo de software, hay un acuerdo común en que el riesgo siempre implica dos características[5]:

- Incertidumbre: El acontecimiento que caracteriza al riesgo puede o no ocurrir; por ejemplo, no hay riesgos de un 100 por ciento de probabilidad.
- Pérdida: Si el riesgo se convierte en una realidad, ocurrirán consecuencias no deseadas o pérdidas.

Cuando se analizan los riesgos es importante analizar el nivel de incertidumbre y el grado de pérdidas asociado con cada riesgo. Para hacerlo, se consideran diferentes categorías de riesgos. Los riesgos del proyecto amenazan al plan del proyecto, es decir, si los riesgos del proyecto se hacen realidad, es probable que la planificación temporal del proyecto se retrase y que los costos aumenten.

Los riesgos del proyecto identifican los problemas potenciales de presupuesto, planificación temporal, personal (asignación y organización), recursos, clientes y requisitos y su impacto en un proyecto de software. Los riesgos técnicos amenazan la calidad y la planificación temporal del software que hay que producir. Si un riesgo técnico se convierte en realidad, la implementación puede llegar a ser difícil o imposible.

Los riesgos técnicos identifican problemas potenciales de diseño, implementación, interfaz, verificación y mantenimiento. Además las ambigüedades de especificaciones, incertidumbre técnica, técnicas anticuadas y las tecnologías de punta son también factores de riesgo. Los riesgos técnicos ocurren porque el problema es más difícil de resolver de lo que pensábamos.

Los riesgos del negocio amenazan la viabilidad del software a construir. A menudo ponen en peligro el proyecto o el producto. Los candidatos para los principales riesgos del negocio son [18]:

- Construir un producto o sistema excelente que no quiere nadie en realidad (riesgo de mercado).
- Construir un producto que no encaja en la estrategia comercial general de la compañía (riesgo estratégico).
- Perder el apoyo de una gestión experta debido a cambios de enfoque o a cambios de personal (riesgo de dirección).
- Perder presupuesto o personal asignado (riesgos de presupuesto).

Es importante recalcar que no siempre funciona una categorización tan sencilla. Algunos riesgos son simplemente imposibles de predecir.

Los riesgos conocidos son todos aquellos que se pueden descubrir después de una cuidadosa evaluación del plan del proyecto del entorno técnico y comercial en el que se desarrolla el proyecto y otras fuentes de información fiables,[18] ejemplo: fechas de entrega poco realistas, falta de especificación de requisitos o de ámbito del software, o un entorno pobre de desarrollo. Los riesgos predecibles se extrapolan de la experiencia en proyectos anteriores, ejemplo: cambio de personal, mala comunicación con el cliente, disminución del esfuerzo del personal a medida que atienden peticiones de mantenimiento, pueden ocurrir pero son extremadamente difíciles de identificar por adelantado.

2.1.1.2 Identificación del riesgo

La identificación del riesgo es un intento sistemático para especificar las amenazas de la planificación del proyecto (estimaciones, planificación temporal, carga de recursos). Identificando los riesgos

conocidos y predecibles, el gestor del proyecto da un paso adelante para evitarlos cuando sea posible y controlarlos cuando sea necesario.[18]

Existen dos tipos diferentes de riesgos: genéricos y específicos del producto.

Los riesgos genéricos son una amenaza potencial para todos los proyectos de software. Los específicos del producto sólo los pueden identificar los que tienen una clara visión de la tecnología, el personal y el entorno específico del proyecto en cuestión. Para identificar los riesgos específicos del producto se da una respuesta a la siguiente pregunta: ¿Qué características especiales de este producto pueden estar amenazadas por el plan del proyecto?

La clave del éxito está en adelantarse a los problemas y tener acciones contempladas para evitar que sucedan o disminuir su impacto y tanto los riesgos genéricos como los específicos del producto se deberían identificar sistemáticamente, porque: "Si no atacas activamente a los riesgos, ellos te atacarán activamente a ti".[19]

Un método para identificar riesgos es crear una lista de comprobación de elementos de riesgo. La lista de comprobación se puede utilizar para identificar riesgos definidos al inicio para estimar un buen método.

La lista de comprobación de factores de riesgo puede organizarse de diferentes maneras, respondiendo a cuestiones relevantes de cada una de los factores de riesgo para estimar, permitiendo valorar su impacto. Finalmente, se lista un conjunto de "componentes y controladores del riesgo" junto con sus probabilidades de aparición. Los controladores del rendimiento, el soporte, el coste y la planificación temporal del proyecto se estudian como respuesta a las preguntas.

Riesgos del impacto en el negocio.

La siguiente lista de comprobación de elementos de riesgo identifica riesgos genéricos asociados con el impacto en el negocio [18]:

¿Efecto de este producto en los ingresos de la compañía?

¿Viabilidad de este producto para los gestores expertos?

¿Es razonable la fecha límite de entrega?

¿Número de clientes que usarán este producto y la consistencia de sus necesidades relativas al producto?

¿Número de otros productos/sistemas con los que este producto debe tener interoperatividad?

¿Sofisticación del usuario final?

¿Cantidad y calidad de la documentación del producto que debe ser elaborada y entregada al cliente?

¿Costos asociados por un retraso en la entrega?

¿Costos asociados con un producto defectuoso?

Cada respuesta para el producto a desarrollar debe compararse con la experiencia anterior. Si se obtiene una gran desviación del porcentaje o si las magnitudes son similares, pero los resultados anteriores fueron poco satisfactorios, el riesgo es grande.

Riesgos relacionados con el cliente.

No todos los clientes son iguales. “Los clientes tienen diferentes necesidades. Algunos saben lo que quieren; otros saben lo que no quieren. Algunos están deseando saber todos los detalles, mientras que otros se quedan satisfechos con vagas promesas”.[5]

Los clientes tienen diferentes personalidades. Algunos disfrutan siendo clientes (la tensión, la negociación, las recompensas psicológicas de un buen producto).

Otros preferirían no ser clientes en absoluto. Algunos aceptarían felizmente cualquier cosa que se les entregara y le sacarían el mejor provecho a un producto pobre. Otros se quejarán amargamente cuando les falte calidad; algunos darán las gracias cuando la calidad es buena; otros se quejarán por todo.

Los clientes se contradicen a menudo. Quieren todo para ayer y gratis. A menudo, el producto se ve atrapado entre las propias contraindicaciones del cliente.

Un "mal" cliente puede tener un profundo impacto en la habilidad del equipo de software para completar el proyecto a tiempo, dentro del presupuesto y también representa una amenaza significativa al plan del proyecto.

La siguiente lista de comprobación de elementos de riesgo identifica riesgos genéricos asociados con diferentes clientes [18]:

¿Ha trabajado con el cliente anteriormente?

¿Tiene el cliente una idea formal de lo que se requiere? ¿Se ha molestado en escribirlo?

¿Aceptará el cliente gastar su tiempo en reuniones formales de requisitos para identificar el ámbito del proyecto?

¿Está dispuesto el cliente a establecer una comunicación fluida con el desarrollador? ¿Está dispuesto el cliente a participar en las revisiones?

¿Está dispuesto el cliente a dejar a su personal hacer el trabajo? ¿Resistirá la tentación de mirar por encima del hombro durante el trabajo técnico?

¿Entiende el cliente el proceso del software?

Si la respuesta a alguna de estas preguntas es "no", se debería hacer una investigación más profunda para valorar el potencial de riesgo.

Riesgos del proceso.

Si el proceso del software no está bien definido; si el análisis, diseño y pruebas se realizan sobre la marcha; si la calidad es un concepto que todo el mundo estima importante, pero por la que nadie actúa de manera tangible para alcanzarla, entonces el proyecto está en peligro.[18]

Aspectos del proceso:

¿Se ha desarrollado en la organización una descripción escrita del proceso del software a emplear en este proyecto?

¿Están de acuerdo los miembros del personal con el proceso del software tal y como está documentado y están dispuestos a usarlo?

¿Se emplea este proceso del software para otros proyectos?

¿Se ha proporcionado una copia de los estándares de ingeniería del software publicados a cada desarrollador y gestor de software?

2.1.1.3 Estimación del riesgo

La estimación del riesgo, intenta medir cada riesgo de dos maneras. La probabilidad de que el riesgo sea real y las consecuencias de los problemas asociados con el riesgo, en caso de ocurrir. El jefe del proyecto, junto con otros gestores y personal técnico, realiza cuatro actividades de proyección del riesgo [18]:

1. Establecer una escala que refleje la probabilidad percibida del riesgo.

2. Definir las consecuencias del riesgo.

3. Estimar el impacto del riesgo en el proyecto y en el producto.

4. Apuntar la exactitud general de la proyección del riesgo de manera que no haya confusiones.

Una tabla de riesgo (Tabla 1) le proporciona al proyecto una sencilla técnica para la estimación del riesgo.

Las categorías para cada uno de los cuatro componentes de riesgo: rendimiento, soporte, coste y planificación temporal, son promediados para determinar un valor general de impacto. A continuación se ilustra una tabla de riesgo como ejemplo [18]:

Tabla 1. Desarrollo de una tabla de riesgo

Riesgos	Categorías	Probabilidad	Impacto
La estimación del tamaño puede ser significativamente baja.	PS	60%	2
Mayor número de usuarios de los previstos.	PS	30%	3
Menos reutilización de la prevista.	PS	70%	2
Los usuarios finales se resisten al sistema.	BU	40%	3
La fecha de entrega estará muy ajustada.	BU	50%	2
Se perderán los presupuestos.	CU	40%	1
El cliente cambiará los requisitos.	PS	80%	2
La tecnología no alcanzará las expectativas	TE	30%	1
Falta de formación en las herramientas.	DE	80%	3
Personal sin experiencia.	ST	30%	2
Habrá muchos cambios de personal.	ST	60%	2

Valores de Impacto	
1	Catastrófico
2	Crítica
3	Marginal
4	Despreciable

Una vez que se han completado las cuatro primeras columnas de la tabla de riesgo, la tabla es ordenada por probabilidad y por impacto. Los riesgos de alta probabilidad y de alto impacto pasan a lo alto de la tabla, y los riesgos de baja probabilidad caen a la parte de abajo. Esto consigue una priorización del riesgo de primer orden.

Los problemas que ocurren durante la estimación de software pueden estar relacionados por uno o varios riesgos. Lo fundamental es determinar “el origen” (qué riesgos ocasionan tal problema) a lo largo de todo el proyecto.

Descripción de los métodos de estimación:

Para la descripción de los métodos de estimación se deben tener en cuenta parámetros como los tiempos a emplear y el esfuerzo de cada rol, en dependencia de la complejidad de los casos de uso, esta se determina de manera empírica pues en la universidad no se cuenta con una basta experiencia en las estimaciones de proyectos.

Roles

Entre los principales roles que intervienen en el método de estimación encontramos el Experto Funcional, Programador Junior, el Analista Diseñador y el Realizador, cada uno de ellos tienen diferentes desempeños en todas las actividades a desarrollar dentro de las etapas.[20]

-Experto Funcional: Es el encargado de aclarar todas las dudas que surjan del negocio a automatizar durante el levantamiento de requisitos y el resto del desarrollo, participará en las pruebas de calidad que se realicen.

-Programador Junior: A este rol se le asignan las tareas de codificación que poseen menor complejidad dentro del proyecto.

-Analista Diseñador: Será el que participa en la definición del proyecto, interviene en la modelación del negocio, interactúa con el usuario final en la definición de los requisitos de la aplicación, crea el modelo de casos de uso del sistema, define el prototipo de interfaz de usuario elemental, es el responsable de traducir la comunicación entre usuarios finales y desarrolladores.

-Realizador: Construye estáticamente el prototipo de interfaz de usuario a partir de las pautas definidas para el diseño de la interfaz, las vistas estáticas que constituyen el punto de partida de los programadores de Interfaz de Usuario.

2.2 Análisis de las métricas existentes

Actualmente cuando se habla de implantar métricas en los diferentes proyectos productivos son muchos los aspectos que hay que tener en cuenta, muchas veces se centran en conocer el tamaño de lo que producen, otras veces el tiempo de desarrollo de software, y otras es necesario calcular la productividad del software, pero en realidad todos estos aspectos son de gran importancia y están estrechamente relacionados.

Las métricas de software aportan una manera de estimar la calidad de los atributos internos del producto, permitiendo así al ingeniero de software valorar la calidad antes de construir el producto, así el tiempo invertido será identificando, examinando y administrando el riesgo, este esfuerzo merece la pena implementarlo por muchas razones ya que habrá disminución de disturbios durante el proyecto, así mismo se podrá desarrollar una habilidad de seguimiento y control del proyecto y se alcanzará la seguridad que da planificar los problemas antes de que ocurran, además conseguiremos absorber una cantidad significativa del esfuerzo en la planificación del proyecto.

Ninguna discusión de la selección y el diseño de métricas de software estarían completos sin abordar el tema de cómo las métricas afectan a las personas y viceversa.

Aunque sea indiscutible, la utilidad de las métricas para la organización, siempre dependerán de las actitudes de las personas involucradas. 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. Estas preocupaciones son reales, y sobreponerse a ellas, requiere un entendimiento de las mediciones, así como saber cómo usar los resultados de las mediciones apropiadamente en todos los niveles de la organización. 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:

• **No mida a los individuos:** El estado del arte en las métricas de software no llega hasta ese punto. El ejemplo clásico de este error es medir la productividad de los individuos. Si se mide la productividad, en líneas de código (LOC) por horas, puede ocurrir que las personas se concentren en su propio trabajo en detrimento del equipo y del proyecto. Hay que enfocarse en los procesos y en los productos, no en las personas.

• **Nunca usar las métricas como un "garrote":** La primera vez que se usen las métricas en contra de los individuos o los equipos será la última vez que se obtendrán datos válidos.

• **No ignorar los datos:** Una forma segura de matar un programa de métricas es ignorar los datos cuando se toman decisiones.

• **Nunca usar una sola métrica:** Los software son complejos y multifacéticos. Un programa de métricas debe reflejar esa complejidad. 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. Enfocarse en una única métrica puede causar que el atributo que es medido mejore a costa de otros atributos.

• **Proveer retroalimentación:** Proporcionando una retroalimentación regular al equipo sobre los datos que ellos ayudan a coleccionar tiene varios beneficios, por ejemplo; ayuda a mantener el foco en la necesidad de coleccionar 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, y 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.

• **Lograr "pertenencia":** Para lograr un sentido de pertenencia tanto en las metas como en las métricas en un programa de medición, se tiene que lograr la participación en la definición de las métricas.

• **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.

1. Individual

2. Equipo de proyecto

3. Organización

A partir de lo anteriormente descrito se puede proseguir al análisis y selección de las diferentes métricas existentes.

1. Interpretación de las mediciones
    - Diferencia entre contextos de pruebas y de uso.
    - Validez de resultados: procedimientos, fuentes de evaluación, validación de datos.
    - Equilibrio de recursos de medición.
    - Especificación correcta.
2. Validación de las métricas
    - Propiedades deseables: confiable, repetible, reproducible, disponible, indicable, correcta, con significado.
    - Demostración de validez: correlación, rastreo, consistencia, predictibilidad, discriminación.
    - 7 propiedades deseables en las métricas
3. Uso de métricas para estimación y predicción
4. Detección de desviaciones y anomalías
5. Presentación de resultados de medición
    - Gráficas de barras, matriz de desempeño, gráficas de Pareto, gráficas de correlación, etc.

Organizadas por característica y subcaracterística, cada métrica contiene:

- Nombre
- Propósito
- Método de aplicación
- Medidas, fórmula y cómputo de datos
- Interpretación del valor medido
- Tipo de escala
- Tipo de medida
- Fuente de medición
- Audiencia

2.2.1 Métricas para la estimación de proyectos en la universidad

Sobre la base de que el uso de las métricas resulta importante para el control de factores decisivos de una organización productora de software con calidad, y partiendo de que en la UCI los conocimientos y aplicaciones de las métricas resultaba en un primer momento un comenzar difuso, se realiza una encuesta (Anexo1) con el objetivos de conocer el nivel de experiencia de aplicación de métricas en los proyectos de la universidad, conocer las métricas que se aplican y las que se necesitan en dependencia de las necesidades de los involucrados en el proceso de desarrollo.

De manera general los resultados de las encuestas realizadas revelan los siguientes datos:

•Algunos proyectos no tienen una idea clara de lo que son las mediciones, ni de qué realmente necesitan medir y ni qué herramientas utilizar para almacenar los indicadores.

• La mayoría de los proyectos conoce la existencia de las métricas pero no para que se utilizan cada una de ellas.

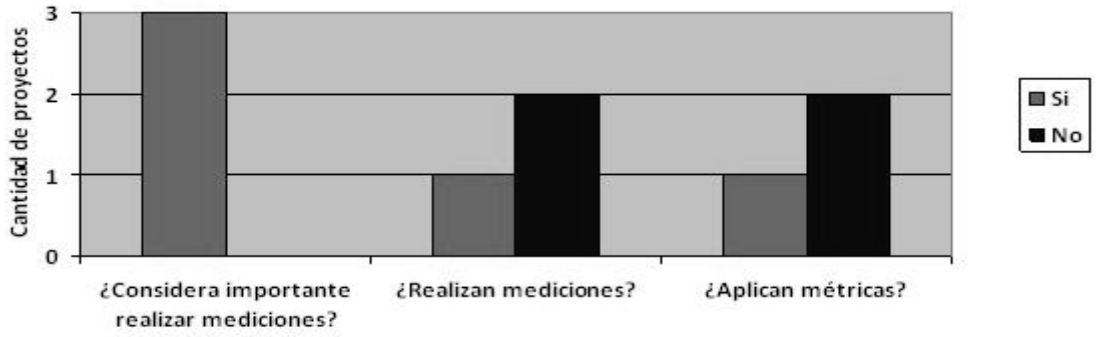
Los puntos más demostrativos que revelan la verdadera situación de los proyectos en cuanto a las métricas se muestran en la siguiente tabla:

Tabla 2. Resultados de la encuesta realizada.

Preguntas	¿Considera importante realizar mediciones?		¿Realizan mediciones?		¿Utilizan métricas?	
Proyectos	Si	No	Si	No	Si	No
CICPC	X		X		X	
INFODREZ	X			X		X
MENPET	X		X		X	

Para una mejor comprensión de los resultados de la tabla 2 se muestra la siguiente gráfica de barras:

Tabla 3. Resultados de la encuesta realizada a través de una gráfica de barras.



De los proyectos que se aplicó la encuesta CICPC fue el que presentó una alta cultura en el tema de las mediciones y métricas.

En base a los resultados obtenidos y analizados la propuesta de solución que se define está pensada para que sea simple de aplicar, teniendo en cuenta que el no conocimiento de métricas en los diferentes proyectos productivos equivale a decir que no existe cultura sobre el uso de medición , que a su vez puede ser un trastorno en la aplicación de una métrica, este proceso requiere de un cambio cultural hacia el control de las actividades y puede ser malinterpretada como pérdida de tiempo, o tal vez mucho control de los desarrolladores lo que provocaría la irritación de los mismos.

2.3 Métricas a proponer

A lo largo de la investigación se han identificado problemas existentes en la Universidad de las Ciencias Informáticas (UCI) los que afectan la eficiencia, la calidad, el tiempo de desarrollo de un producto, la productividad y la efectividad en el desarrollo de los diferentes proyectos de software. No existen conocimientos claros acerca de las mediciones que deben hacerse a los proyectos. En este aspecto es bueno aclarar que no solo se desea medir para tener un control previo del proyecto, sino

que resulta muy necesario medir para tener una idea exacta de cómo va evolucionando este durante todo su ciclo de vida.

El resultado arroja que la solución desde el punto de vista teórico está en aplicar las métricas que se proponen dado que permiten que los proyectos de software tengan una estrategia bien definida, así como una buena planificación del trabajo y una buena organización de la producción desde los proyectos y hacia la alta directiva para gestionar los compromisos.

Las métricas a proponer miden el desarrollo de los proyectos y ayudan a los líderes y desarrolladores de los mismos en la toma decisiones y acciones correctivas, así como el mejoramiento continuo del proceso de desarrollo del software, obteniéndose mejores resultados.

Uno de los aspectos a medir es el tiempo, la importancia de medir este indicador ha transcendido por cada uno de los diferentes proyectos y se ha evidenciado serias dificultades del avance del software en cuanto al tiempo, es decir, no siempre se tiene una idea clara y exacta del desarrollo del proyecto en cuanto al tiempo, de ahí que se propongan métricas a aplicar con el fin de mejorar este aspecto.

2.3.1 Métricas de Tiempo de desarrollo de Software

La primera métrica a proponer es la métrica de tiempo de desarrollo de software. El tiempo representa una métrica base y necesaria de medir, pues influye sobre otras métricas derivadas donde funciona relacionada con otro atributo de medición.

Como se explicó anteriormente en la actualidad la necesidad de medir el desarrollo de un proyecto de software en cuanto al tiempo es imprescindible, existe la necesidad de entregar proyectos de software en el tiempo preciso.

Las métricas de tiempo permiten calcular el tiempo real dedicado por la persona en cada una de las tareas ejecutadas como parte del proyecto y son expresadas en horas. A continuación se brinda un ejemplo de una métrica de tiempo.

Tabla 4. Ejemplo de métrica de tiempo para un proyecto de software.

Nombre	**Métrica de tiempo**
Propósito	Cuál es el tiempo real para completar una tarea.
Método de aplicación	Evaluar la eficiencia de las llamadas al Sistema Operativo y a la aplicación Estimar el tiempo basado en ello. Puede medirse: • Todo o parte de la especificación de diseño.

	• Probar la ruta completa de una transacción. • Probar módulos o partes completas del producto. • Producto completo durante la fase de prueba.
Medición , Fórmula	T = cantidad de horas de duración de un proyecto (calculado o simulado).
Interpretación	Entre más corto mejor.
Tipo de escala	Proporción
Tipo de medida	T = tiempo(horas)
Fuente de medición	Sistema Operativo conocido. Tiempo estimado en llamadas al sistema.
Audiencia	Desarrolladores

2.3.1.1 Tiempo Estimado

El tiempo estimado es el mismo tiempo real que en un momento adecuado se está utilizando como umbral para estimar otros proyectos y a partir de estas estimaciones realizar las planificaciones reales obteniendo nuevos tiempos reales que en su debido momento actualizarán los umbrales.

2.3.1.2 Error Estimado de Tiempo

Esta métrica permite apreciar el margen de error en la estimación de tiempo. Constituye un indicador que muestra la calidad en la estimación de tiempo.

Entre más cercano a 0 sea este valor, mejor será nuestra estimación de tiempo. Valores sustancialmente mayores que 0 indican que el tiempo real está siendo sustancialmente mayor que el tiempo estimado y la estimación está siendo demasiado irreal. Valores sustancialmente menores que 0 indican que nuestra estimación está siendo muy conservadora.

La fórmula que permite expresar esta métrica es la siguiente:

EET = ([T] - [TE]) / [TE]

Donde:

EET: Error Estimando Tiempo

T: Tiempo

TE: Tiempo Estimado

2.3.2 Métricas de Costo de desarrollo de Software.

Para calcular el costo de desarrollo de software los principios son semejantes a los cálculos para el tiempo de desarrollo de software; una métrica base que igualmente relacionada con otros atributos formarán métricas derivadas.

Existen dos factores fundamentales a examinar en la estimación de un proyecto de software, su duración y costo, la importancia es cada vez mayor de mantener una idea exacta del costo de un proyecto como factor estratégico.

Se ha determinado que sea el costo de un proyecto uno de los aspectos más prioritarios en su realización.

El costo de un proyecto de software da la posibilidad de conocer cuánto se gasta en el proyecto y que es lo que realmente se necesita para obtener un software con calidad y con la menor cantidad de recursos posibles.

Las estimaciones de costos son necesarias para establecer un presupuesto para el proyecto o para asignar un precio para el software de un cliente.

Las métricas de costo propuestas a continuación dan un resultado de cómo calcular el costo en cualquier proyecto de software, y de manera particular en la universidad, puesto que la UCI como productora de software debe tener una idea exacta de los recursos necesarios para la realización de los proyectos, es decir debe controlar la cantidad de computadoras que utilizan y el capital imprescindible. Las métricas de costo para la UCI son importantes para nutrir el método de estimación en la contratación de los proyectos tanto nacionales como de exportación.

2.3.2.1 Razón de Costo de Planificación

Otra forma de medir la calidad de la planificación es a través del análisis de la **razón de costo de planificación (RCP)**. La RCP es la razón del costo planificado, dividido entre el costo real hasta la fecha en que se calcule.

En caso de que dicha razón se igual a 1, significa que se gasta en planificar lo mismo que se gasta en desarrollar, o sea, el costo no varía.

En caso de que la RCP sea menor que 1, se evidencia pobre desempeño del proyecto, se gasta más de lo que se gana. Si el análisis es de tiempo, puede ser que se esté gastando mucho tiempo en planificar los proyectos.

Idealmente la RCP debe ser ligeramente mayor que 1, lo cual demostraría un buen desempeño del proyecto, o sea, se gasta menos de lo que se gana.

También se puede plantear que esta métrica indica la calidad de la planificación. Esta vez indicando el grado en que se están cumpliendo las metas planificadas. Idealmente la RCP debe ser 1. Preferiblemente mayor que 1, para contar con una holgura de tiempo moderada para posibles riesgos. Si es menor que 1 es que se está gastando más tiempo que lo planificado en los proyectos. Si son substancialmente mayores que 1 los planes están siendo muy conservadores.

La fórmula que permite describir lo anterior se muestra a continuación:

RCP = [TE] / [T]

Donde:

RCP: Razón de Costo de Planificación

TE: Tiempo Estimado

T: Tiempo real

Como se puede apreciar es necesario tener conocimiento de las métricas de tiempo y tiempo estimado para lograr un resultado en la métrica de razón de costo de planificación.

2.3.3 Métricas de Tamaño de desarrollo de Software.

El Tamaño de desarrollo de software del proyecto, es otro factor importante que puede afectar la precisión de las estimaciones, es una métrica base pues influye sobre otras métricas derivadas donde funciona relacionada con otro atributo de medición. A medida que el tamaño aumenta, crece rápidamente la interdependencia entre varios elementos del software como por ejemplo la complejidad en el desarrollo.

De manera general el tamaño del software se expresa en LOC (líneas de código). Esta medida por supuesto sería la más óptima, pero teniendo en cuenta que se está proponiendo medir el tamaño de los productos UCI dependiendo de sus características, entonces resulta necesario detenerse a analizar y llegar a la conclusión de en qué medir los proyectos UCI, utilizando la unidad de medida más genérica y que acople a todas las existentes.

Medidas de tamaño existente en la UCI:

1. Guiones: Un guión es un caso de uso, en dependencia de la complejidad se clasifica en Alta, Media y Baja

- Guión técnico: Explica detalladamente qué contiene cada pantalla.
- Guión de contenido: Lo hace el cliente o pedagogo, por ejemplo explicando el por qué y para qué se hace la pantalla.

2. POP: Medida utilizada en los proyectos de diseño. Son todos los materiales creados en un proyecto de diseño.
3. Requisitos: La relación de varios requisitos pueden conformar un caso de uso.
4. Historia de usuario: Una historia de usuario es un caso de uso. Para la clasificación del caso de uso hay que extraer del contenido de la descripción, la cantidad de Flujos Básicos y Flujos Alternos e incluirlo en la clasificación de la complejidad de estos según la cantidad. La idea principal es describir un caso de uso en dos o tres líneas con la terminología del cliente (de hecho, se supone que deben ser escritos por el mismo), de tal manera que se creen test de aceptación para el usuario y permita hacer una estimación de tiempo de desarrollo del mismo.
5. BPMN: Se basa en la descripción de procesos. Un proceso puede estar formado por uno o varios casos de uso. Todo depende de la complejidad.
6. Módulos: Un módulo está compuesto por varios sub módulos y a la vez estos están contenidos por uno o varios casos de uso relacionados que se clasificarán en dependencia de la complejidad.

Teniendo en cuenta este análisis se determina que todas estas medidas pueden convertirse en casos de uso como unidad de medida estándar para la UCI.

2.3.3.1 Tamaño

Esta métrica brinda el tamaño real. Se expresa en casos de uso.

2.3.3.2 Tamaño Estimado

El tamaño estimado es el mismo tamaño real que en un momento determinado va a servir como base de partida para realizar estimaciones en otros proyectos. Se expresa en CU (Casos de Uso).

2.3.3.3 Error Estimado del Tamaño

Permite apreciar el margen de error en la estimación de tamaño. Esta métrica constituye un indicador de nuestra calidad en la estimación de tamaño.

Entre más cercano a 0 sea el valor mejor será nuestra estimación de tamaño. Valores sustancialmente mayores que 0 indican que el tamaño real esta siendo sustancialmente mayor que el tamaño estimado y nuestra estimación está siendo demasiado irreal. Valores sustancialmente menores que 0 indican que nuestra estimación está siendo muy conservadora.

2.3.4 Métricas de productividad de desarrollo de Software

En la actualidad, cuando se habla de implantar métricas de productividad en las empresas donde se produce software se centran en conocer el tamaño de lo que se produce, en este caso se refiere al software que se desarrolla. Para calcular la productividad, se puede hacer uso de una métrica base:

- Las métricas orientadas al tamaño.

2.3.4.1 Métricas orientadas al tamaño

Es para saber en qué tiempo se va a terminar el software y cuántas personas se van a necesitar. Son medidas directas al software y el proceso por el cual se desarrolla, si una organización de software mantiene registros sencillos.

Las métricas del software orientadas al tamaño provienen de la normalización de las medidas de calidad y/o productividad considerando el "tamaño" del software que se haya producido. Si una organización de software mantiene registros sencillos, se puede crear una tabla de datos orientados al tamaño, que lista cada proyecto de desarrollo de software y las medidas correspondientes de cada proyecto.

Las métricas orientadas al tamaño no están aceptadas universalmente como el mejor modo de medir el proceso de desarrollo del software. La mayor parte de la discusión gira en torno al uso de las líneas de código pero en nuestra universidad se usa las mediciones por casos de uso.

2.3.4.2 Productividad

El término productividad se desarrolla de manera diferente en las distintas bibliografías consultadas, muchas veces se refieren a la producción de código por hora y en otras ocasiones al esfuerzo necesario para desarrollar un software, en el caso de la universidad este criterio está basado a partir de los casos de uso, en sentido general se puede definir como la relación existente entre el tamaño del software por unidad de tiempo o esfuerzo realizado.

Las métricas de productividad recogen la eficiencia del proceso de producción y relacionan el software que se ha construido con el esfuerzo que ha costado elaborarlo, y además 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.

La métrica de productividad propuesta expresa la productividad real, o sea, la cantidad de casos de uso que se producen en una hora.

Se expresa en CU/h.

La fórmula para calcular la productividad real es:

P= t / T

Donde:

P: Productividad

t: Tamaño

T: Tiempo

La tendencia debe ser que la productividad vaya aumentando paulatinamente.

2.3.4.3 Productividad Estimada

Esta métrica cumple con las mismas características que la anterior, solo que esta vez se va analizar la productividad antes de comenzar el proyecto, es decir, se va estimar el posible valor de la producción del software en cuanto al tamaño y el tiempo, a partir de ahí se comenzará a trabajar en aras de cumplir con las expectativas de esta estimación.

Esta métrica expresa la productividad estimada, o sea, la cantidad estimada de casos de uso que se producen en una hora.

Se expresa en CU/h.

La fórmula para calcular la productividad es la siguiente:

PE=te/ TE

Donde:

PE: Productividad Estimada

te: Tamaño Estimado

TE: Tiempo Estimado

La tendencia debe ser que la productividad vaya aumentando paulatinamente.

2.3.5 Métricas de Calidad de Software.

La mejor forma de planificar con exactitud es medir y darle seguimiento a los procesos de desarrollo y trabajar para mejorarlos. Esto se puede lograr siguiendo un proceso de planificación consistente,

analizando los datos históricos y planificando entonces al detalle. También es importante aclarar que durante el desarrollo del proceso de control de configuración, muchos de los cambios que son solicitados son producto de defectos en el software o el prototipo entregado al usuario final, por esta razón es importante controlar los defectos que aparecen de esta manera en el producto. Una forma de analizar el estado de los defectos es a través del resumen de defectos, donde se analizan la densidad de defectos por etapas, y los escapes netos. Se considera escapes netos a todos los defectos que se insertaron antes o durante una etapa determinada, pero que no fueron encontrados en esa etapa, sino en otra etapa posterior. Todo lo anterior se puede controlar mediante métricas de calidad, las cuales son propuestas a continuación.

2.3.5.1 Densidad de Defectos

Revela la cantidad real de defectos que se eliminan por cada caso de uso. La tendencia en esta métrica es ir disminuyendo paulatinamente.

La fórmula para calcular esta métrica es la siguiente:

DD= DR / t

Donde:

DD: Densidad de Defectos

DR: Defectos Removidos (defectos eliminados reales)

t: Tamaño

2.3.5.2 Densidad de Defectos Estimados

Esta métrica revela cuáles son los posibles defectos que se pueden detectar en el software, simplemente es una estimación, pero no deja de ser importante, puesto que evita males mayores y ayuda a erradicar los defectos antes que estos ocurran. Ofrece la cantidad estimada de defectos que se eliminan por cada caso de uso. La tendencia en esta métrica es ir disminuyendo paulatinamente los defectos.

La fórmula para calcular la densidad de defectos estimada es:

DDE =DRE / te

Donde:

DDE: Densidad de Defectos Estimados

DRE: Defectos Removidos Estimados (estimación de los defectos a eliminar)

te: Tamaño Estimado

2.3.6 Métricas de eficiencia

La eficiencia es de suma importancia para obtener productos de alta calidad en una empresa de software, pero en la universidad no son todos los proyectos los que ya han medido por esto se proponen estas métricas con el objetivo de medir la eficiencia en los proyectos específicos que puedan utilizarla.

2.3.6.1 Tiempo de respuesta

Esta métrica responde a cuánto tiempo toma completar una tarea en particular y cuánto toma antes de que el sistema responda a determinada operación.

Comenzar una tarea específica. Medir el tiempo que esta toma en completar una muestra hasta que concluya la operación emprendida. Guardar los registros de cada intento.

Se calcula como: Tiempo empleado en obtener el resultado.

0 < T A mayor prontitud (menor tiempo) resultará mejor.

2.3.6.2 Tiempo de espera

Esta métrica responde a qué proporción de tiempo los usuarios dedican a esperar por una respuesta del sistema.

Ejecutar los casos de pruebas en situaciones características y tareas concurrentes. Medir el tiempo que toma completar las operaciones seleccionadas. Guardar un registro de cada intento y calcular el tiempo medio para cada caso o situación.

Se calcula como: [X =Ta/Tb]

Donde:

X: Tiempo de espera

Ta: Tiempo total dedicado a esperar.

Tb: Tiempo consumido por la tarea.

0 <= X Cuanto menor resultará mejor

2.3.7 Métrica de Esfuerzo

Esta métrica se utiliza para que el jefe del equipo tenga más elementos a la hora de asignar, reasignar carga de trabajo y analizar el esfuerzo del personal bajo su mando.
Se calcula como:
E=T * H
Donde:
E: Esfuerzo
T: Tiempo
H: Hombres

2.3.8 Métricas de Prueba

La etapa de prueba es muy importante puesto que en ella se puede probar la calidad con que se ha desarrollado el software. En esta etapa no se puede asegurar la ausencia de defectos solo puede demostrarse la existencia de los mismos. En todas las fases del desarrollo del proyecto hay que probar el software que se va construyendo y resulta muy importante definir un conjunto de mediciones para guardar los resultados de las mismas de manera que aporten información relevante para pruebas sucesivas.

2.3.8.1 Índice de Madurez

El estándar IEEE 982.1-1988 sugiere la métrica del índice de madurez del software (IMS). Esta métrica proporciona una indicación de la estabilidad del producto de software basada en los cambios que ocurren con cada versión del producto.

La ecuación para obtener los resultados de esta métrica es la siguiente:
ISM= [Mt – (Mc + Ma + Md)]/ Mt]
Donde:
ISM: Índice de Madurez
Mt: Número de módulos en la versión actual
Mc: Número de módulos en la versión actual que han cambiado
Ma: Número de módulos en la versión actual que se han añadido
Md: Número de módulos en la versión actual que se han eliminado

2.3.8.2 Tasa de Defectos

Esta métrica es importante que la utilicen los grupos de calidad de cada proyecto pues proporciona la tasa de defectos.

La Tasa de Defectos se calcula mediante la fórmula:

TD= D/CU

Donde:

TD: Tasa de Defectos

D: Número de Defectos

CU: Cantidad de Casos de Uso

2.3.8.3 Tasa de Errores

Esta métrica brinda la tasa de errores que no es más que los errores que se producen por casos de uso.

La Tasa de Errores se calcula mediante la siguiente ecuación:

TasE=E/CU

Donde:

TasE: Tasa de Errores

E: Número de errores

CU: Cantidad de Casos de Uso

2.3.8.4 Tiempo Real de Revisión

Esta métrica calcula el tiempo real dedicado a una revisión según la complejidad planificada de un caso de uso. Se expresa en CU/horas.

El tiempo real de revisión está dados por:

TRR= CU/ ER

Donde:

TRR: Tiempo Real de Revisión

CU: Cantidad de Casos de uso

ER: Esfuerzo empleado en la revisión

Esta métrica será tanto para:

- Pruebas Funcionales: Estimación inicial
- Documentación: Estimación según su complejidad por capítulos y por casos de uso
- Complejidad Alta: Un capítulo que tenga más de 15 páginas.

- Complejidad Media: Un capítulo que tenga entre 8 y 14 páginas.
- Complejidad Baja: Un capítulo que tenga menos de 8 páginas.
- El tipo de capítulo que más predomine en el documento pasará a ser la clasificación de la complejidad del documento.

Ej. Un documento de 5 capítulos. 2 de complejidad alta, 1 medio y 1 bajo entonces el documento tiene una clasificación de Alto

En caso de coincidir la misma cantidad por clasificación entonces el documento se clasifica por exceso.

Ej. Un documento de 3 capítulos. Cada capítulo Alto, Medio y Bajo entonces el documento tiene una clasificación de Alto.

En la siguiente tabla se puede apreciar la estimación por tipo de revisión.

Tabla 5. Estimación por tipo de revisión

Tipo de Revisión	Complejidad Alta	Complejidad Media	Complejidad Baja
Manual Instalación	2 horas	1 horas	0,5 horas
Manual de Ayuda			
Especificación de requerimiento	A valoración en cualquiera de las dos clasificaciones		
Doc. Arquitectura	3 horas	2 horas	1 horas
Doc. Despliegue			
Doc. Arquitectura de Información			

2.3.9 Satisfacción del cliente

Las métricas que se proponen a continuación tienen como objetivo cumplir con las especificidades del cliente, muestran un conjunto de requisitos, todos en aras de que el producto resultante cuente con la calidad requerida.

2.3.9.1 Tasa de Estabilidad

Esta métrica proporcionará a los proyectos una medida de la estabilidad que se va alcanzando en las diferentes etapas de desarrollo del software.

La Tasa de Estabilidad está dada por:
TEb=1-(NC/CU)
Donde:
TEb: Tasa de Estabilidad

NC: Número de cambios
CU: Cantidad de Casos de Uso

Proporciona un indicador de cómo una aplicación cumplió las expectativas del cliente o usuario.

El número de cambios son los cambios solicitados durante el primer trimestre (90 días) después de la implementación.

La cantidad de CU es el tamaño de la aplicación en Casos de Uso.

2.3.9.2 Fiabilidad

Las métricas de fiabilidad son unidades de medida de fiabilidad del sistema. La fiabilidad se mide para contar el número de caídas operacionales y, donde es apropiado, relacionando esto a las demandas hechas en el sistema y el tiempo que el sistema ha estado funcionando, además se requiere de un programa de medición a largo plazo para evaluar la fiabilidad de sistemas críticos.

La fórmula para calcular esta métrica se muestra a continuación:

F=1-(NF/CU)
Donde:
F:Fiabilidad
NF:Número de fallos
CU:Cantidad Casos de Uso
El número de fallos se va a tener en cuenta desde que se puso en marcha o se le entregó al cliente la aplicación, incluyendo además todos los posibles fallos que puedan surgir repetidamente.
Los CU es el total de casos de uso de la aplicación que esté siendo medida. Este cálculo debe realizarse mensual o trimestralmente.

2.3.9.3 Cumplimiento de la planificación de entrega

Esta métrica considera los compromisos de entrega del producto final o por etapas cuantificando la desviación en los plazos de entrega en tiempo.

CPE= (TR * 100)/ TE
Donde:
CPE: Cumplimiento de la planificación de entrega.
TR: Tiempo Real que se dedicó realmente a la entrega del producto.

TE: Tiempo Estimado o planificado, fue el que se pactó con el cliente desde su inicio.

El resultado de esta métrica indica:

>100%: hay retrasos en el plazo de entrega

≤100%: se están consiguiendo mejor los resultados que lo planificado.

Este cálculo debe realizarse mensual o trimestralmente.

2.3.10 Métricas para el Aseguramiento de la Calidad

El Aseguramiento de la Calidad consiste en tener y seguir un conjunto de acciones planificadas y sistemáticas, implantadas dentro del Sistema de Calidad de cualquier empresa y específicamente de nuestra universidad. Estas acciones deben ser demostrables para proporcionar la confianza adecuada (tanto a la propia empresa como a los clientes) de que se cumplen los requisitos del Sistema de la Calidad. Además son un conjunto de actividades para evaluar el proceso mediante el cual se desarrolla el producto.

Con las métricas propuestas se espera que se lleve un control exacto del avance de estas actividades en la universidad, con el objetivo de que el producto cumpla con las especificidades del cliente y con los requisitos que plantea el Sistema de Calidad.

2.3.10.1 Esfuerzo dedicado a las actividades de SQA

Esta métrica es de suma importancia para cumplir los requisitos impuestos por el sistema de calidad, puesto que mide el esfuerzo dedicado a cada una de las actividades de aseguramiento de calidad, es decir, se pretende medir que es lo que realmente se ha hecho para cumplir con las expectativas del cliente.

El Esfuerzo dedicado a las actividades de SQA está dado por:

ESQA = h/H

Donde:

ESQA: Esfuerzo dedicado a las actividades de SQA

h: Horas

H: Hombres

2.3.10.2 Tiempo dedicado al SQA

Esta métrica calcula el tiempo real dedicado a cada una de las actividades de aseguramiento de la calidad.

La ecuación para obtener el tiempo dedicado al SQA es:

TSQA= ESQA/ET

Donde:

TSQA: Tiempo dedicado al SQA

ESQA: Esfuerzo dedicado a las actividades de SQA

ET: Esfuerzo total de proyecto

2.3.10.3 Cumplimiento de las actividades de SQA

Esta métrica es para verificar el cumplimiento de cada una de las actividades de aseguramiento de la calidad.

La fórmula para obtener resultados en esta métrica es la siguiente:

CSQA=NR/NP

Donde:

CSQA: Cumplimiento de las actividades de SQA

NR: Número de revisiones, pruebas y auditorias realizadas

NP: Número de revisiones, pruebas y auditorias planeadas

2.3.10.4 Eficiencia y efectividad

La eficiencia es el grado en que optimiza los recursos de cómputo, es decir, permite medir hasta que punto son eficientes cada una de las computadoras que se utilizan en la realización de un proyecto.

La fórmula para estas métricas es:

NC= NCSR / (NCR + NCSR).

Donde:

NCSR: Número de no conformidades sin resolver elevadas al líder de proyecto

NCR: Número de no conformidades reportadas

A continuación se muestra una tabla resumen de las métricas anteriormente propuestas:

Tabla 6. Resumen de las métricas propuestas

	Métricas	Fórmula	Descripción
Tiempo	[Tiempo]		Tiempo real dedicado por la persona en cada una de las tareas ejecutadas como parte del proyecto. Se expresa en horas.
	[Tiempo Estimado]		Tiempo estimado por la persona en cada una de las tareas que debe ejecutar como parte del proyecto. Se expresa en horas.
	[Error Estimando Tiempo]	EET=[(T - TE)] / [TE] EET: Error Estimando Tiempo TE: Tiempo Estimado T : Tiempo real	Permite apreciar el margen de error en la estimación de tiempo. Entre más cercano a 0 sea este valor, mejor será nuestra estimación de tiempo. Valores sustancialmente mayores que 0 indican que el tiempo real está siendo sustancialmente mayor que el tiempo estimado y nuestra estimación está siendo demasiado irreal. Valores sustancialmente menores que 0 indican que nuestra estimación está siendo muy conservadora.
Costo	[Razón del Costo de Planificación(RCP)]	RCP=[TE] / [T] RPC: Razón del costo de planificación TE: Tiempo Estimado T : Tiempo real	Métrica que indica calidad de la planificación. Si es menor que 1 es que se está gastando más tiempo que lo planificado en los proyectos. Si son substancialmente mayores que 1 los planes están siendo muy conservadores.
Tamaño	[Tamaño Estimado]		Tamaño estimado Se expresa en CU (Casos de uso).
	[Tamaño]		Tamaño real Se expresa en CU (Casos de uso).
	[Error Estimando Tamaño]	EEt=([t] - [te]) / [te] EEt: Error estimado tamaño t: Tamaño te: Tamaño Estimado	Permite apreciar el margen de error en la estimación de tamaño. Esta métrica constituye un indicador de nuestra calidad en la estimación de tamaño. Entre más cercano a 0 sea el valor mejor será nuestra estimación de tamaño. Valores sustancialmente mayores que 0 indican que el tamaño real esta siendo sustancialmente mayor que el tamaño estimado y nuestra estimación está siendo demasiado irreal. Valores sustancialmente menores que 0 indican que nuestra estimación está siendo muy conservadora.
Esfuerzo	[Esfuerzo]	[E=T*H] E:Esfuerzo T:Tiempo H:hombre	Se utiliza para que el jefe del equipo tenga más elementos a la hora de asignar, reasignar carga de trabajo y analizar el esfuerzo del personal bajo su mando.

Productividad	[Métricas orientadas al tamaño]		Se utiliza para ver en que tiempo va a terminar el software y cuantas personas se van a necesitar
	[Productividad]	P=[t] / [T] P:Productividad t:Tamaño T: Tiempo	Expresa la productividad real, o sea, la cantidad real de casos de uso que se producen en una hora. Se expresa en LOC/h. La tendencia debe ser que la productividad vaya aumentando paulatinamente.
	[Productividad Estimada]	PE= [te] / [TE] P:Productividad Estimada t:Tamaño Estimado T: Tiempo Estimado	Expresa la productividad estimada, o sea, la cantidad estimada de casos de uso que se producen en una hora. Se expresa en CU/h. La tendencia debe ser que la productividad vaya aumentando paulatinamente.
Calidad	[Densidad Defectos]	DD=[DR] / [t] DD: Densidad Defectos DR: Defectos Removidos t: Tamaño	Revela la cantidad real de defectos que se eliminan por líneas de código. La tendencia en esta métrica es ir disminuyendo paulatinamente
	[Densidad Defectos Estimada]	DDE=[DRE] /te] DDE: Densidad Defectos Estimada DRE: Defectos removidos Estimados te: Tamaño Estimado	Revela la cantidad estimada de defectos que se eliminan por líneas de código. La tendencia en esta métrica es ir disminuyendo paulatinamente.
Eficiencia	[Tiempo de respuesta]	[Tiempo empleado en obtener el resultado]	Esta métrica responde a cuanto tiempo se tomará en completar una tarea en particular y cuanto toma antes de que el sistema responda a determinada operación. El resultado indica que a menor tiempo resultará mejor.
	[Tiempo de espera]	[X=Ta/Tb] X:Tiempo de espera Ta: Tiempo total dedicado a esperar Tb: Tiempo consumido por la tarea	Esta métrica responde a que proporción del tiempo los usuarios dedican a esperar por una respuesta del sistema. El resultado indica que cuanto menor sea X resultará mejor.
Prueba	[Índice de madurez]	ISM=[Mt-(Mc+Ma+Mb)/Mt] ISM: Índice de madurez Mt: Número de módulos en la versión actual Mc: Número de módulos en la versión actual que han cambiado Ma: Número de módulos en la versión actual que se han añadido Mb: Número de módulos en la versión actual que se han eliminado	Proporciona una indicación de la estabilidad del producto de software basada en los cambios que ocurren en cada versión del producto.

	[Tasa de defectos]	TD=D/CU TD: Tasa de Defectos D: Número de Defectos CU: Cantidad de Casos de Uso	Los defectos por casos de uso
	[Tasa de errores]	TasE=E/CU TasE: Tasa de errores E: Número de errores CU: Cantidad de Casos de Uso	Los errores por casos de uso
	[Tiempo Real de Revisión]	TRR=CU/ER TRR: Tiempo Real de Revisión CU: Cantidad de Casos de Uso ER: Esfuerzo empleado en la revisión	Calcula el tiempo real dedicado a una revisión según la complejidad planificada de un caso de uso. Esta métrica será tanto para: • Pruebas funcionales • Documentación: Estimación según la complejidad por capítulos -Complejidad Alta: Un capítulo con mas de 15 páginas. -Complejidad Media: Un capítulo que tenga entre 8 y 14 páginas. -Complejidad Baja: Un capítulo que tenga menos de 8 páginas.
Satisfacción del cliente	[Tasa de Estabilidad]	TEb=1-(NC/CU) TEb: Tasa de Estabilidad NC: Número de cambios CU: Cantidad de casos de uso	Esta métrica proporcionará a los proyectos una medida de la estabilidad que se va alcanzando en las diferentes etapas de desarrollo del software.
	[Fiabilidad]	F=1-(NF/CU) F:Fiabilidad NF: Número de fallos CU: Cantidad de Casos de Uso	Los números de fallos por casos de uso.
	[Cumplimiento de la planificación de entrega]	CPE=(TR*100)/TE CPE: Cumplimiento de la planificación de entrega TR: Tiempo Real que se dedico realmente a la entrega del producto TE: Tiempo Estimado, fue el que pacto con el cliente desde su inicio	Considera los compromisos de entrega del producto final o por entrega cuantificando la desviación en los plazos de entrega en tiempo. El resultado de esta métrica indica: >100%: hay retrasos en el plazo de entrega ≤100%: se están consiguiendo mejor los resultados que lo planificado.

Aseguramiento de la Calidad	[Esfuerzo dedicado a las actividades de SQA]	ESQA=h/H ESQA: Esfuerzo dedicado a las actividades de SQA h: Horas H: Hombre	Mide el esfuerzo dedicado el esfuerzo dedicado a cada una de las tareas del aseguramiento de la calidad.
	[Tiempo dedicado a las SQA]	TSQA=ESQA/ET TSQA: Tiempo dedicado a las SQA ESQA: Esfuerzo dedicado a las actividades de SQA ET: Esfuerzo total del proyecto	Calcula el tiempo real dedicado a cada una de las actividades de aseguramiento de la calidad.
	[Cumplimiento de las actividades SQA]	CSQA=NR/NP CSQA: Cumplimiento de las actividades SQA NR: Número de revisiones, pruebas y auditorias realizadas. NP: Número de revisiones, pruebas y auditorias planeadas.	Verifica el cumplimiento de cada una de las actividades de aseguramiento de la calidad.
	[Eficiencia y efectividad]	NC=NCSR/(NCR+NCSR) NCSR: Número de no conformidades sin resolver elevadas al líder de proyecto NCR: Número de no conformidades reportadas	Permite medir hasta que punto son eficientes las computadoras que se utilizan en la realización de un proyecto.

El creciente desarrollo de la Industria de Software ha traído consigo la necesidad de producir software con calidad, y para lograrlo se tienen en cuenta numerosos factores, entre los que se encuentran las métricas de software, una herramienta indiscutible para ayudar a mantener el control de los procesos y productos durante el desarrollo del software.

Capítulo 3: Validación de la Solución Propuesta

En este capítulo se realizará la evaluación técnica de la propuesta descrita en el capítulo anterior. Se usará el método multicriterios para dicha evaluación, el cuál se basa en la evaluación cuantitativa de criterios previamente definidos por parte de expertos en el tema. Por lo que se describirá la forma de aplicar este método y los elementos necesarios para el mismo, posteriormente se presentarán los resultados obtenidos de la evaluación.

3.1 Método para la validación de la propuesta

Para validar la propuesta se utilizó el método de experto, que permite tomar decisiones para aceptar o no la propuesta de acuerdo con los criterios definidos.[21]

Para llevar a cabo el desarrollo del mismo se efectuaron un conjunto de pasos:

1. Se elaboran los criterios de evaluación de acuerdo a las características de la propuesta y se organizan por grupos.

Grupo No.1: Criterios de mérito científico

- Valor científico de la propuesta.
- Calidad de la investigación.
- Aporte científico.
- Novedad científica.

Grupo No.2: Criterios de implantación

- Satisfacción de las necesidades de los ingenieros de software.
- Necesidad del empleo de la propuesta.
- Solución del problema existente para realizar las estimaciones de los futuros proyectos productivos en la universidad.

Grupo No.3: Criterios de flexibilidad

- Adaptabilidad a proyectos de software.
- Uso de las métricas necesarias para la estimación de proyectos de software.
- Los resultados de las métricas aplicadas son de fácil interpretación.

Grupo No.4: Criterios de impacto

- Repercusión en los proyectos productivos.
- Aceptación de la propuesta.
- Posibilidades de aplicación.
- Impacto en el área a la cual está destinada.

2. Se le asigna un peso relativo a cada grupo de criterios de acuerdo al porcentaje que representa cada grupo del total y los intereses a evaluar.

Grupo No.1…… 25

Grupo No.2…… 30

Grupo No.3……...20

Grupo No.4……...25

3. Se organiza un comité de expertos con una cantidad mínima de 7 teniendo en cuenta su especialidad, grado científico y currículo.

4. Se les entrega a los expertos la propuesta para que estudien el tema a evaluar y dos modelos, uno para que valorar el peso relativo de cada criterio *(Ver Anexo 2)* y otro para realizar una evaluación cuantitativa de cada criterio con una escala de 1-5 y la apreciación cualitativa con una clasificación final del proyecto en excelente, bueno, aceptable, cuestionable y malo. También se da la posibilidad de dar su opinión haciendo una valoración final del proyecto, emitiendo todas aquellas consideraciones que estimaron convenientes *(Ver Anexo 3)*.

5. Después de recibir los valores del peso relativo de cada criterio se construye la Tabla 7

• Sea C el número de criterios que van a evaluarse y E el número de expertos que realizan la evaluación.

Tabla 7.Resultado del trabajo de expertos.

G	C/E	E1	E2	E3	E4	E5	E6	E7	Ep
25	C1								
	C2								
	C3								
	C4								
30	C5								
	C6								
	C7								
20	C8								
	C9								
	C10								
25	C11								
	C12								
	C13								
	C14								
T									

6. Se verifica la consistencia en el trabajo de los expertos, para lo que se utiliza el coeficiente de concordancia de Kendall y el estadígrafo Chi cuadrado (X^2). Se sigue el procedimiento siguiente:

• Para cada criterio se determina:

∑E: Sumatoria del peso dado por cada experto

Ep: Puntuación promedio del peso dado por cada experto

M∑E: media de los ∑E

ΔC: Diferencia entre ∑E y M∑E

• Se determina la desviación de la media, que posteriormente se eleva al cuadrado para obtener la dispersión (S) por la expresión.

$$S = \sum (\sum E - \sum \sum E / C)^2$$

• Conociendo la dispersión se puede calcular el coeficiente de concordancia de Kendall (W).

$$W = S / E^2 (C^3 – C)/ 12$$

• El coeficiente de concordancia de Kendall permite calcular el Chi cuadrado real.

$$X^2 = E (C-1) W$$

• Los valores obtenidos se muestran en la Tabla 8.

Tabla 8. Tabla para el cálculo de concordancia de Kendall.

Expertos/Criterios	**E1**	**E2**	**E3**	**E4**	**E5**	**E6**	**E7**	**∑E**	**Ep**	**ΔC**	**ΔC^2**
C1								0	0	0	0
C2								0	0	0	0
C3								0	0	0	0
C4								0	0	0	0
C5								0	0	0	0
C6								0	0	0	0
C7								0	0	0	0
C8								0	0	0	0
C9								0	0	0	0
C10								0	0	0	0
C11								0	0	0	0
C12								0	0	0	0
C13								0	0	0	0
C14								0	0	0	0
DC	0	0	0	0	0	0	0	0	0	0	0
M∑E	0										
W	0										
X^2	0										

• El Chi cuadrado calculado se compara con el obtenido de las tablas estadísticas.

• Si se cumple:

$$X^2 real < X^2 (\alpha, c-1)$$

Existe concordancia en el trabajo de expertos.

7. Si no existe concordancia se hace necesario repetir el trabajo de expertos.

8. Después de comprobar la consistencia del trabajo de expertos se puede definir el peso relativo de cada criterio (P).

9. Conociendo el peso de cada criterio y la calificación dada por los evaluadores en una escala de 1-5 se puede construir la Tabla 9, para obtener el valor de P × c., donde (c), es el criterio promedio concebido por los expertos.

Tabla 9. Tabla de calificación de cada criterio.

Criterios	Calificación (c)					P	P x c
	1	2	3	4	5		
C1							
C2							
C3							
C4							
C5							
C6							
C7							
C8							
C9							
C10							
C11							
C12							
C13							
C14							

10. Se calcula el índice de aceptación del proyecto (IA).

$$IA = \Sigma (P \times c) / 5$$

11. Por último se determina la probabilidad de éxito de la propuesta.

Rangos predefinidos de Índice de Aceptación.

IA > 0,7 Existe alta probabilidad de éxito

0,7 > IA > 0,5 Existe probabilidad media de éxito

0,5 > IA > 0,3 Probabilidad de éxito baja

0,3 > IA Fracaso seguro

Por lo que la probabilidad de éxito es:

3.2 Análisis de la evaluación técnica de la propuesta

En el análisis de la evaluación técnica de la propuesta se utilizaron 7 expertos para que dieran su opinión y valoraran dicha propuesta. Primeramente los expertos emitieron su juicio para darle peso a cada criterio con la cual se elaboró la tabla de los valores de peso relativo de cada criterio *(Ver Anexo4)*.

Luego se llevaron los valores de la tabla para el cálculo de concordancia entre los expertos *(Ver Anexo5)*.

El resultado de los cálculos fueron los siguientes:

X^2 real es 15,4609, para seleccionar el X^2 de la tabla se toma $1-\alpha = 0.99$, dónde α es el error permisible, entonces α= 0.01. Debe cumplirse que $X^2 < X^2$ (α, c-1).

De esta forma quedaría:

$15{,}4609 < 27{,}6882$ por lo que se puede afirmar que existe concordancia entre los expertos, por lo que se puede pasar a construcción de la tabla de clasificación de cada criterio para saber el índice de aceptación de la propuesta *(Ver Anexo 6)*.

Después de tener todos los datos en la tabla se calcula el valor del Índice de Aceptación (IA) que sería:

0,75228, se compara el valor con los valores que aparecen a continuación para saber la valoración de la propuesta.

IA > 0,7 Existe alta probabilidad de éxito
0,7 > IA > 0,5 Existe probabilidad media de éxito
0,5 > IA > 0,3 Probabilidad de éxito baja
0,3 > IA Fracaso seguro

Por lo que la probabilidad de éxito es alta.

En este capítulo se utilizó el método multicriterio para determinar si la propuesta es viable. Se analizó el resultado de aplicar dicho método, en el cual se obtuvo una probabilidad de éxito alta, indicando que

la aplicación de la propuesta proporcionará resultados favorables y que lo planteado hasta el momento brinda un aporte significativo, capaz de resolver los problemas existentes por los que se inició la investigación.

Conclusiones

Después de haber realizado una investigación profunda de las métricas para estimar los proyectos de software y la situación actual en la que se encuentra la UCI al respecto, se logró realizar la propuesta de métricas para estimar los proyectos de software de la UCI, basado en:

- Aplicabilidad a todos los proyectos de software de la universidad, tanto para los que han medido, como para los que aún no han recopilado medidas.
- Definición de métricas objetivas en dependencia de las necesidades de información de la universidad.
- Definición de métricas fáciles de entender y usar, equilibrando la aceptación de todos proyectos de software, para una primera versión.
- Definición de indicadores para las próximas producciones de software en la UCI.
- Obtención de registros históricos para estimar los proyectos de software en la UCI.

Finalmente se hizo la evaluación técnica de la propuesta, arrojando que la probabilidad de éxito es alta, lo que implica desde el punto de vista teórico, que si se proponen métricas para estimar el proceso de desarrollo de los proyectos de software en la universidad, se espera que se eleve la calidad del software que se produce.

Recomendaciones

Se recomienda:

- Aplicar la propuesta en los proyectos de software de la universidad.
- Actualizar las métricas de software propuestas a medida que avance el proceso de medición en los proyectos y surjan nuevas necesidades de información.
- Automatizar las métricas de software propuestas para un fácil manejo de las mismas.
- Comunicar estos resultados a la dirección de IP.
- Utilizar el trabajo de diploma como material de estudios para investigaciones posteriores.

Referencias Bibliográficas

1. ALVAREZ. 2003 [cited; Available from: http://alvarez.jc.googlepages.com/PanormicaEFQM_paginadoactas_.pdf.
2. INFOCALIDAD. 2005 [cited; Available from:http://www.infocalidad.net/gest_calidad_def/definicion.asp.
3. SOFTWARE, I.D. 2005 [cited; Available from: http://www.monografias.com/trabajos15/ingenieria-software/ingenieria-software.shtml.
4. MARTÍNEZ, R.D., *ConfigCASE 3.0 Herramienta de apoyo a la Gestión de apoyo a la gestión de configuración. Propuesta Arquitectónica., Instituto Superior Politécnico "José Antonio Echeverría".* 2006.
5. PRESMAN, R.S., *"Ingeniería del Software. Un Enfoque Práctico".* 1998.
6. Fenton, S.L.P.y.N.E., *Software metrics. A rigorous and practical approach.* 1997.
7. Len, E.O. 1991 [cited; Available from: http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/gonzalez_d_h/capitulo2.pdf
8. Brian. 1996 [cited; Available from: http://www.alarcos.inf-cr.uclm.es/doc/Calidad/capitulo09.ppt.
9. Somerville, *Ingenieria de Software.* 2002.
10. RUP,"*Rational Unified Process".* 2003.
11. CAPUCHINO, A.M.M.S., *"Control y gestión de proyectos software, Unidad 4: Estimación de Proyectos Software".* 1996.
12. MARÍA, M.S.C.A., *"Estimación de Proyectos Software".* 1998.
13. HURTADO, L.L., *"La primera comunidad libre donde aprender y compartir".* 2007.
14. ESCORIAL, J.S., *"Calidad de Software: Medidas del Proceso".* 2006.
15. HERNÁNDEZ, S.E.B., *"Métricas de estimación de tamaño: Puntos de Caso de Uso".* 2002.
16. WIKIPEDIA1, *"Técnicas de estimación y seguimiento".* 2006.
17. MARCELO, J., *"Técnicas de estimación y seguimiento".* 2006.
18. MENÉNDEZ, R., *"Gestión de riesgos en ingeniería del software".* 2004.
19. BASTERRA, R.V.D., *"Administrando la inseguridad".* 2006.

20. ARIAS, O.Y.A., *"Proyecto Técnico de Asesoría Especializada, Colaboración Médica Odontológica, Comunicación Institucional y Solución Tecnológica para apoyar la modernización del Sistema Penitenciario de la República Bolivariana de Venezuela".* 2006.

21. GOLIATH, K.D. and Y.R. MARTINEZ, *Documentación imprescindible para los flujos de trabajo de diseño e implementación de software de gestión*. 2007.

Bibliografía Consultada

1. **Autores, Colectivo de.** [En línea] 2001. [Citado el: 15 de enero de 2008.] http://www.di.uniovi.es/~cueva/publicaciones/2001/MetricasEnTiempoReal.pdf.

2. **Alvarez.** [En línea] 2003. http://alvarez.jc.googlepages.com/PanormicaEFQM_paginadoactas_.pdf.

3. **Brian.** [En línea] 1996. http://www.alarcos.inf-cr.uclm.es/doc/Calidad/capitulo09.ppt.

4. **Específica, Seminario de formación.** Calidad de software. [En línea] 2006. http://www.calidaddelsoftware.com/modules.php?name=News&file=article&sid=183.

5. **Giraldo, Otoniel Perez.** [En línea] http://www.willydev.net/descargas/willydev_planeasoftware.pdf.

6. **Infocalidad.** [En línea] 2005. http://www.infocalidad.net/gest_calidad_def/definicion.asp.

7. **J, Nieves Sosa E y Álvarez Rodríguez F.** [En línea] 2004. http://ingsw.ccbas.uaa.mx/pagSER/ObjAprend/publicaciones/VeranoCiencia2004.pdf.

8. **Len.** [En línea] 1991. http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/gonzalez_d_h/capitulo2.pdf

9. **Mendosa, Gonzalo Mena.** [En línea] 2006. http://www.mena.com.mx/gonzalo/maestria/calidad/presenta/iso_9126-3/.

10. **Teleformación**. [En línea] http://teleformacion.uci.cu/.

11. **Tesis**. [En línea] http://tesis.uci.cu/.

Anexo1. Cuestionario de Métricas de Software

Este cuestionario tributará a la tesis: “Propuesta de métricas para la estimación de proyectos”. Para realizar el cuestionario no necesita dar constancia de su nombre, los autores les brindan confidencialidad.

Datos de la Persona Entrevistada

1. Facultad a la que pertenece *(marque con una x)*:

   1 __ 2__ 3 __ 4__ 5__ 6 __ 7 __ 8__ 9__ 10 __

2. ¿Qué rol desempeña?

   _____

Datos del Proyecto

3. Proyecto al que pertenece:
4. ¿Cuántos años lleva usted desarrollando software?

_____

Datos sobre medición en los proyectos:

1. ¿Considera usted importante realizar mediciones? *(marque con una x)*

   ____Sí ____No

2. ¿En su proyecto se realizan mediciones? *(marque con una x)*

   ____Sí

   ____No

   ____No sé

En caso de que su respuesta sea que sí responda.

3. ¿Cuál de las herramientas utiliza en su proyecto para guardar las medidas?

   ___Project Planning

   ___Process Dashboard

   ___Team Process Dashboard

___PSM Inside

___Excel

4. ¿En su proyecto, qué le dificulta medir? *(marque con una x)*

__ El tamaño del software

__ La complejidad del software

__ El esfuerzo del personal

__ Tiempo de desarrollo de las actividades

__ La calidad del software

__ Otras. Mencione cuáles

5. ¿En su proyecto, qué le interesa medir? *(marque con una x)*

__La productividad

__El tamaño del software

__La complejidad del software

__El esfuerzo del personal.

__Tiempo de desarrollo de las actividades

__La calidad del software

__Otras. Mencione cuáles

Datos específicos del uso de métricas del proyecto

1. ¿Se utilizan métricas en su proyecto?

Si___ No___

2. Considera usted importante aplicar métricas en su proyecto para medir el *(marque con una x)*:

___Índice de Madurez

___Tiempo de Reparación

___Tasa de Defectos

___Tasa de Errores

___Integridad de la descripción del producto

___Efectividad

___Esfuerzo

___Efectividad

___Cumplimiento

___Complejidad

___Estabilidad

___Fiabilidad

3. ¿Conoce o ha utilizado estos métodos de estimación en su Proyecto?

Métodos de	Conoce	No	Utiliza	No
Juicio del				
Método				
COCOMO.				
Puntos de				
COCOMO II				
Otro(s) que				

Anexo 2.Guía para informar el peso de los criterios

Modelo No. 1

Guía para informar el peso de los criterios.

Fecha de recepción _____

Fecha de entrega _____

Nombre y Apellidos del evaluador _____

Le otorgará un peso a cada criterio de acuerdo a su opinión y el peso total de cada grupo debe sumar:

Grupo No.1…… 25

Grupo No.2…… 30

Grupo no.3…… 20

Grupo No.4……..25

Para que el peso total asignado sea 100.

Grupo No. 1: Criterios de mérito científico

1. Valor científico de la propuesta.

   Peso.....

2. Calidad de la investigación.

   Peso.....

3. Aporte científico.

   Peso.....

4. Novedad científica.

   Peso.....

Grupo No. 2: Criterios implantación

5. Satisfacción de las necesidades de los ingenieros de software.

   Peso.....

6. Necesidad del empleo de la propuesta.

   Peso.....

7. Solución del problema existente para realizar las estimaciones de los futuros proyectos productivos en la universidad.

   Peso……..

Grupo No.3: Criterios de flexibilidad

8. Adaptabilidad a proyectos de software.

   Peso.....

9. Uso de las métricas necesarias para la estimación de proyectos de software.

   Peso.....

10. Los resultados de las métricas aplicadas son de fácil interpretación.

    Peso……

Grupo No.4: Criterios de impacto

11. Repercusión en los proyectos productivos.

    Peso.....

12. Aceptación de la propuesta.

    Peso.....

13. Posibilidades de aplicación.

    Peso.....

14. Impacto en el área a la cual está destinada.

    Peso.....

Anexo 3. Guía para la evaluación

Modelo No. 2

Guía para la evaluación.

Fecha de recepción _____

Fecha de entrega _____

Nombre y Apellidos del evaluador_____

- Criterios de medida que se evalúan en una escala de 1 - 5

Grupo No. 1: Criterios de mérito científico

1. Valor científico de la propuesta.

    Peso.....

2. Calidad de la investigación.

    Peso.....

3. Aporte científico.

    Peso.....

4. Novedad científica.

    Peso.....

Grupo No. 2: Criterios implantación

5. Satisfacción de las necesidades de los ingenieros de software.

Peso.....

6. Necesidad del empleo de la propuesta.

Peso.....

7. Solución del problema existente para realizar las estimaciones de los futuros proyectos productivos en la universidad.

Peso……..

Grupo No.3: Criterios de flexibilidad

8. Adaptabilidad a proyectos de software.

Peso.....

9. Uso de las métricas necesarias para la estimación de proyectos de software.

Peso.....

10. Los resultados de las métricas aplicadas son de fácil interpretación.

Peso……

Grupo No.4: Criterios de impacto

11. Repercusión en los proyectos productivos.

Peso.....

12. Aceptación de la propuesta.

Peso.....

13. Posibilidades de aplicación.

Peso.....

14. Impacto en el área a la cual está destinada.

Peso.....

• Categoría final del proyecto

___ Excelente: Alta novedad científica, con aplicabilidad y resultados relevantes.

___ Bueno: Novedad científica, resultados destacados.

___ Aceptable: Suficientemente bueno con reservas.

___ Cuestionable: No tiene relevancia científica y los resultados son mal.

___ Malo: No aplicable.

- Valoración final

Sugerencias del evaluador para mejorar la calidad del proyecto

Elementos críticos que deben mejorarse.

Anexo 4. Tabla de los valores del peso relativos a cada criterio.

G	C/E	E1	E2	E3	E4	E5	E6	E7	Ep
25	**C1**	5	6	6	7	5	6	7	6,000
	C2	8	9	6	8	8	7	8	7,714
	C3	6	5	7	5	5	6	6	5,714
	C4	6	5	6	5	7	6	4	5,571
30	**C5**	8	9	10	10	10	10	11	9,714
	C6	15	14	10	10	10	10	10	11,286
	C7	7	7	10	10	10	10	9	9,000
20	**C8**	6	6	8	6	5	6	6	6,143
	C9	7	7	6	7	9	6	7	7,000
	C10	7	7	6	7	6	8	7	6,857
25	**C11**	8	7	7	5	5	7	5	6,286
	C12	5	6	6	5	5	6	5	5,429
	C13	6	6	6	8	7	6	8	6,714
25	**C14**	6	6	6	7	8	6	7	6,571
T		100	100	100	100	100	100	100	99,999

Anexo 5. Tabla para el cálculo de Concordancia.

Expertos/Criterio	E1	E2	E3	E4	E5	E6	E7	∑E	Ep	ΔC	ΔC^2
C1	5	6	6	7	5	6	7	42	6,000	8	64
C2	8	9	6	8	8	7	8	54	7,714	4	16
C3	6	5	7	5	5	6	6	40	5,714	10	100
C4	6	5	6	5	7	6	4	39	5,571	11	121
C5	8	9	10	10	10	10	11	68	9,714	18	324
C6	15	14	10	10	10	10	10	79	11,286	29	841
C7	7	7	10	10	10	10	9	63	9,000	13	169
C8	6	6	8	6	5	6	6	43	6,143	7	49
C9	7	7	6	7	9	6	7	49	7,000	1	1
C10	7	7	6	7	6	8	7	48	6,857	2	4
C11	8	7	7	5	5	7	5	44	6,286	6	36
C12	5	6	6	5	5	6	5	38	5,429	12	144
C13	6	6	6	8	7	6	8	47	6,714	3	9
C14	6	6	6	7	8	6	7	46	6,571	4	16
DC	100	100	100	100	100	100	100	700	99,999	128	1894
M∑E	50										
W	0,1699										
X^2	15,4609										

Anexo 6. Tablas para la calificación de cada criterio.

Criterios	Calificación (c)					P	P x c
	1	2	3	4	5		
C1			X			0,06000	0,18000
C2				X		0,07714	0,30856
C3			X			0,05714	0,17142
C4			X			0,05571	0,16713
C5				X		0,09714	0,38856
C6				X		0,11286	0,45144
C7				X		0,09000	0,36000
C8				X		0,06143	0,24572

C9				X		0,07000	0,28000
C10				X		0,06857	0,27428
C11				X		0,06286	0,25144
C12				X		0,05429	0,21716
C13				X		0,06714	0,26856
C14			X			0,06571	0,19713
Total							3,7614
IA	0,75228						

Glosario de términos

Calidad de software: Concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados, y con las características implícitas que se espera de todo software desarrollado profesionalmente.

CMM: Es un estándar en ingeniería de software que hace hincapié en la mejora del proceso de software en base a los procedimientos internos y sin descuidar a las personas.

CMMI: Modelo para la mejora o evaluación de los procesos de desarrollo y mantenimiento de sistemas y productos de software.

Estimación: Es una pequeña planeación sobre que es lo que va a ser en un proyecto.

Indicador: Es una métrica o combinación de métricas que proporcionan una visión

del proceso, del proyecto o del software en sí, y poder hacer ajustes para que las cosas

mejoren.

Medición: Acto de determinar una medida.

Medida: Proporciona una indicación cuantitativa de extensión, cantidad, dimensiones, capacidad y tamaño de algunos atributos de un proceso o producto.

Métrica: Es una medida cuantitativa del grado en que un sistema o proceso posee un atributo dado.

TIC: Tecnologías de la Información y las Comunicaciones.

Planificación: Es la actividad fundamental del gestor de proyecto que comprende la formulación de lo que hay que realizar para obtener una finalidad que será precisamente la del sistema que estamos planificando (Decidir + Hacer).Control: Es inspección, fiscalización, intervención.

Procesos: Atributos de actividades relacionadas con el software.

Productividad: Es la cantidad de esfuerzo requerido para lograr un cierto grado de funcionalidad.