Universidad de las Ciencias Informáticas


Facultad 8

Trabajo de Diploma para optar por el título
de Ingeniero en Ciencias Informática

# Propuesta de métricas para evaluar el flujo de trabajo Análisis y Diseño.

**Autoras:** Adieren Acosta Zamora
Daisel Betancourt Ramírez

**Tutor:** Ing. Osiris Pérez Moya

Ciudad de La Habana

Junio 2008

*Cuando pueda medir lo que está diciendo y expresarlo con números, ya conoce algo sobre ello; cuando no pueda medir, cuando no pueda expresar lo que dice con números, su conocimiento es precario y deficiente; puede ser el comienzo del conocimiento, pero en tus pensamientos apenas estás avanzando hacia el escenario de la ciencia.*

*Lord Kelvin*

# Declaración de Autoría

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

Para que así conste firmamos la presente a los ____días del mes de Junio del año 2008.

Adieren Acosta Zamora

_____

Firma del Autor

Daisel Betancourt Ramírez

_____

Firma del Autor

Ing. Osiris Pérez Moya

_____

Firma del Tutor

# Agradecimientos

La realización de nuestra tesis no hubiese sido posible sin la ayuda, el apoyo y la colaboración incondicional e inquebrantable de aquellos que de una forma u otra nos orientaron en la creación de cada página de este trabajo.

En especial a nuestro tutor el Ing. Osiris Pérez Moya por su dedicación y paciencia.

A la Ing. Dayami Rodriguez Brito por brindarnos su ayuda y a todo el personal profesional de la Infraestructura Productiva de la universidad que leyó nuestra tesis, y que sus comentarios y observaciones nos permitieron enriquecerla.

A todas nuestras compañeras del apto que nos hicieron más amena nuestra estancia en la universidad, gracias por estar con nosotras estos 5 años, compartir risas y llantos en todos los momentos.

A todos los profesores que durante todos estos años contribuyeron con nuestra formación profesional.

A nuestras amistades por confiar siempre en nosotras.

A todos ellos nuestro agradecimiento infinito.

*Daisel*

A mis padres por estar conmigo incondicionalmente, sin sus enseñanzas no estaría aquí ni sería quien soy ahora. Al fin les regalo lo que tanto esperaron. Los quiero mucho.

A mi familia por su interés y preocupación.

A Adieren, mi compañera de tesis porque mejor no la hubiese querido, por su confianza y optimismo, juntas alcanzamos este triunfo.

*Adieren*

A mis padres, por su apoyo e infinito amor, y en especial a mami por ser mi principal fuente de inspiración. Los quiero mucho.

A mi hermana por su eterno cariño y confiar siempre en mí.

A mis abuelos y tías por su preocupación.

A mis padrinos por estar siempre a mi lado y apoyarme tanto. Los quiero mucho.

Especialmente a Yoan mi novio por su dedicación durante estos tres años. Te amo mucho.

A mi compañera de tesis Daisel por su paciencia, confianza y porque juntas logramos hacer realidad nuestro sueño.

## Dedicatoria

*A mis padres por estar siempre unidos,*

*y apoyarme en todo momento.*

*A mi familia y amigos que confiaron siempre en mí.*

*Daisel*

*A mis padres y hermana por estar siempre a mi lado y confiar siempre en mí.*

*A mis padrinos por su apoyo incondicional.*

*A mi novio Yoan por hacerme tan feliz.*

*Adieren*

# Resumen

Hoy en día las compañías de todo el mundo industrializado reconocen que la calidad del producto se traduce en ahorro de costos y en una mejora general. La industria de desarrollo de software no es la excepción, por lo que en los últimos años se han realizado intensos trabajos para aplicar los conceptos de calidad en el ámbito del software. Una de las causas que provoca que el producto no tenga la calidad requerida es el control y evaluación del mismo. Para un correcto control de la calidad se investigan métricas las cuales permiten evaluar de forma estadística elementos de la calidad del software. El objetivo principal del presente trabajo es: realizar una propuesta de métricas que permitan evaluar el flujo de trabajo Análisis y Diseño, en el desarrollo de software que utilizan una programación orientado a objeto en la Universidad de las Ciencias Informáticas. Para lograr lo planteado anteriormente se realizó una investigación detallada sobre los objetivos, las principales características y actividades del flujo de trabajo Análisis y Diseño de la metodología de planificación y desarrollo RUP, se analizaron el conjunto de métricas existentes en el mundo y se adaptaron a este flujo de trabajo, realizando una detallada y fácil interpretación de las mismas conformando la propuesta; posteriormente se efectuó la validación de la misma. La propuesta es aplicable solo a los proyectos que utilicen la metodología RUP.

**Palabras Claves**

Métricas, medición, método de medición, calidad, Análisis y Diseño, metodología RUP.

# Índice

# Índice de Tablas y Figuras

## Índice de Tablas



## Índice de Figuras

# Introducción

La industria de software, a diferencia de otras industrias, tiene muy poco tiempo de existir. Lo que ha llamado la atención del mercado hacia ella han sido dos factores esenciales: la velocidad con que ha crecido y su alcance. Desde su inicio existieron personas en distintos campos que vieron el avance que para ellos representaba hacer uso de software especializado que les permitiera automatizar procesos o acelerarlos.

Al haber tanta demanda en cuanto al campo se iniciaron muchas investigaciones en la rama de software y de hardware. Con el tiempo los costos se redujeron y el software se convirtió en un negocio rentable. Al haber tanto interés, muchas personas empezaron a desarrollar y ahí nacieron las primeras grandes empresas de software.

Esto trajo consigo un problema natural en el proceso: al haber tantos desarrolladores en distintos países y para distintas aplicaciones empezó a haber diversidad de estilos así como la calidad del producto final variaba mucho entre producto y producto. En este marco se hizo necesario un estándar o una metodología que permitiera a los consumidores de software decidir si el producto que estaban recibiendo era de calidad y si cumplía ciertos requisitos de funcionalidad.

Desde la década del 70, la calidad de un producto de software ha sido motivo de preocupación para especialistas, ingenieros, investigadores y comercializadores de software, los cuales han realizado gran cantidad de investigaciones al respecto con dos objetivos fundamentales:

¿Cómo obtener un software con calidad?

¿Cómo evaluar la calidad del software?

A partir de 1996 en Cuba se dan los primeros pasos para el ordenamiento de un trabajo continuo destinado a impulsar el uso y desarrollo de las Tecnologías de la Información y las Comunicaciones y se inicia progresivamente en la creación de software; pero empezó haber una demora en la entrega de los productos por falta de calidad, pues no cumplían con las necesidades o expectativas del cliente, por lo que no se estaba aplicando un estándar que permitiera evaluar la calidad de un software. Esta problemática actualmente es motivo de investigaciones.

La UCI[1] contribuye con la producción de software en nuestro país y comienza a dar sus primeros pasos en la obtención de la calidad mediante metodologías de

[1] UCI: *Universidad de las Ciencias Informáticas*. Centro de formación de profesionales y productor de software mediante la vinculación estudio-trabajo.

planificación y desarrollo. Uno de los estándares que se han venido investigando en este centro son las métricas que permiten evaluar de forma estadística elementos de la calidad del software, por lo que nos propusimos realizar una innovadora investigación a partir de la siguiente *situación problemática:* en la UCI no se han definido un conjunto de métricas que logren retroalimentar estadísticamente la calidad en el flujo de trabajo Análisis y Diseño que propone la metodología RUP[2] para la organización y planificación del software.

La situación planteada anteriormente, nos lleva a definir el siguiente problema científico: ¿cómo evaluar la en el flujo de trabajo Análisis y Diseño propuesto por RUP para el desarrollo del software?

Partiendo del problema científico el *objeto de estudio* es: la evaluación de la calidad del flujo de trabajo Análisis y Diseño, y el campo de acción en el cual se adentra la investigación son las métricas que permitan evaluar el flujo de trabajo Análisis y Diseño propuesto por RUP en los proyectos de la Universidad de las Ciencias Informáticas.

Se define como *objetivo general* de la investigación, realizar una propuesta de métricas que permitan evaluar el flujo de trabajo de Análisis y Diseño, en el desarrollo del software que utilizan una programación orientado a objeto en la Universidad de las Ciencias Informáticas y como *objetivos específicos*:

1. Realizar un estudio sobre la evaluación de la calidad de un software.
2. Analizar con especialistas en la Universidad de las Ciencias Informáticas sobre la calidad de un software.
3. Analizar las métricas existentes en el mundo.
4. Analizar las métricas para valorar cuál de ellas es aplicable al flujo de trabajo de Análisis y Diseño.
5. Elaborar la propuesta de métricas
6. Validar la propuesta con un comité de experto.

La investigación se centra en la siguiente idea a defender: si se proponen métricas para evaluar el flujo de trabajo de Análisis y Diseño que propone RUP para el desarrollo de un software se espera una retroalimentación estadística de evaluación de la calidad del software que se desarrolla en la Universidad de las Ciencias Informáticas.

[2] RUP: *Proceso Unificado del Desarrollo de Software*. Es la metodología más usada en la universidad para organizar y desarrollar software.

Para darle solución a los objetivos trazados, se definen las siguientes tareas de investigación:

1. Investigar en libros u otras fuentes bibliográficas sobre los modelos y estándares que existen en el proceso de evaluación de la calidad de un software.
2. Investigar sobre la evaluación de la calidad del proceso en la producción de software en la Universidad de las Ciencias Informáticas.
3. Realizar una detallada búsqueda sobre las métricas.
4. Investigar con expertos las métricas que se están utilizando en la Universidad de las Ciencias Informáticas.
5. Analizar las métricas para el flujo de trabajo Análisis y Diseño.
6. Realizar la propuesta de métricas que permitan evaluar estadísticamente el flujo de trabajo Análisis y Diseño
7. Validar la propuesta de solución

El contenido de este trabajo está estructurado en tres capítulos, organizados de la siguiente manera:

Capítulo 1: titulado por “Fundamentación teórica”, donde se reflejan las principales definiciones y conceptos relacionados con el tema, además de los diferentes tipos de métricas existentes y se hace una valoración de la evaluación de la calidad en la UCI.

Capítulo 2: titulado por “Propuesta de Solución”, donde primeramente se realiza una descripción detallada de las características que presenta el flujo de trabajo Análisis y Diseño y en lo que nos basamos para realizar la propuesta de solución.

Capítulo 3: titulado por “Validación de la Propuesta”, donde realizamos la validación de la propuesta de solución por el comité de experto seleccionado.

# Capítulo 1
# Fundamentación Teórica

## *1.1 Introducción*

En el mundo actual la evaluación de la calidad del software es un tema bastante polémico, por esta razón, después de haber realizado un análisis general sobre la situación del software y la calidad de los mismos ya fuese a nivel, mundial, en nuestro país y en la UCI, este capítulo ayudará a fundamentar varios conceptos relacionados precisamente con la calidad, su medición y evaluación, por lo que se conceptualiza y se caracteriza los diferentes tipos de métrica. Se explican algunos modelos y estándares de calidad así como la metodología de desarrollo de software que se aplica en los proyectos productivos de la Universidad de las Ciencias Informáticas y si en este centro se están evaluando cuantitativamente estos proyectos productivos.

## *1.2 Calidad de Software*

### 1.2.1 ¿Qué es la calidad del software?

La calidad del software es el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario.[3]

La calidad del software es el conjunto de cualidades que lo caracterizan y que determinan su utilidad y existencia. La calidad es sinónimo de eficiencia, flexibilidad, corrección, confiabilidad, mantenibilidad, portabilidad, usabilidad, seguridad e integridad. [3]

Conjunto de propiedades y de características de un producto o servicio, que le confieren aptitud para satisfacer una necesidad explícita o implícita. [3]

### 1.2.2 ¿Cómo controlar y evaluar la calidad del software?

El control de calidad es el conjunto de técnicas y actividades de carácter operativo, utilizadas para verificar los requerimientos relativos a la calidad del producto o servicio. [3]

Para obtener un software con calidad es necesaria la utilización de metodologías o estándares que permitan equilibrar los procedimientos de trabajo, a favor de lograr una

mayor confiabilidad, mantenibilidad y usabilidad, al mismo tiempo que eleven el rendimiento, tanto para el desarrollo del producto como para el control de la calidad del software.

La calidad de un software se debe controlar y evaluar para garantizar que los desarrolladores elaboren un producto que cumpla con las especificaciones del cliente. Para controlar la calidad es necesario evaluarla y esto implica una comparación entre los requisitos anteriormente establecidos y el producto finalmente desarrollado.[4]

Los ingenieros del software aplican mediciones e indicadores que le permitan evaluar cuantitativamente la calidad de las diferentes actividades en los flujos de trabajo que se han establecido al aplicar la ingeniería del software. Para obtener esta evaluación de calidad, el ingeniero debe utilizar medidas técnicas, que evalúan la calidad con objetividad, no con subjetividad.[4]

Para controlar la calidad del software es necesario, ante todo, definir los parámetros, indicadores o criterios de medición, ya que, como bien plantea Tom De Marco, "  no puede controlar lo que no se puede medir".

Los autores coinciden en que el software posee determinados índices medibles que son las bases para la calidad, el control y el perfeccionamiento de la productividad.[5]

Una vez seleccionados los índices de calidad, se debe establecer el proceso de control, que requiere los siguientes pasos: [5]

- Definir el software que va a ser controlado: clasificación por tipo, esfera de aplicación, complejidad, de acuerdo con los estándares establecidos para el desarrollo del software.
- Seleccionar una medida que pueda ser aplicada al objeto de control. Para cada clase de software es necesario definir los indicadores y sus magnitudes.
- Crear o determinar los métodos de valoración de los indicadores: métodos manuales como cuestionarios o encuestas estándares para la medición de criterios periciales y herramientas automatizadas para medir los criterios de cálculo.
- Definir las regulaciones organizativas para realizar el control: ¿quiénes participan en el control de la calidad?, ¿cuándo se realiza?, ¿qué documentos deben ser revisados y elaborados?

### 1.2.3 Estándares de Calidad

Los estándares de calidad pueden ser utilizados para construir mejores productos y asegurar su calidad. Construir un modelo de calidad es bastante complejo y es usual que estos modelos descompongan la calidad del producto software jerárquicamente en una serie de características y sub-características que pueden usarse como una lista de comprobación de aspectos relacionados con la calidad.

#### 1.2.3.1 Estándar ISO/IEC 9126

La ISO/IEC 9126 es un estándar internacional para la evaluación del software. Está enfocada a la calidad de producto y consta de las siguientes partes: [6]

Parte 1: Modelo de calidad (9126-1)

Parte 2: Métricas Externas (9126-2)

Parte 3: Métricas Internas (9126-3)

Parte 4: Calidad en el uso (9126-4)

Este estándar proviene del modelo establecido en 1977 por McCall y sus colegas, que propusieron un modelo para especificar la calidad del software. A pesar de que es uno de los más antiguos se ha extendido en todo el mundo. [6]

La especificación y la evaluación de la calidad de producto de software se pueden conseguir definiendo características de calidad apropiadas, tomando en cuenta el objetivo de uso del producto de software. [7]

La norma define un modelo de calidad basado en dos partes bien identificadas[8]:

- La calidad interna y externa.
- La calidad de uso.

La calidad interna, entendida como la totalidad de las características del producto software desde un punto de vista interno, y la calidad externa definida como la totalidad de las características de producto software desde un punto de vista externo influyen en la calidad del proceso, al mismo tiempo que la calidad de uso influye sobre las anteriores. La segunda parte del modelo especifica cuatro características de calidad durante el uso del producto

Calidad interna, externa y de uso están relacionadas, una se sustenta en la otra como capas sucesivas. La calidad del proceso influye en la calidad del producto que a su vez es relevante en la calidad de uso.[8]

La calidad de cualquiera de los procesos del ciclo de vida, contribuye a mejorar la calidad del producto, y esta a su vez contribuye a mejorar la calidad en el uso. Por consiguiente, evaluar y mejorar un proceso es un medio para mejorar la calidad del producto; la evaluación y mejora de la calidad del producto son una vía para mejorar la calidad durante el uso (Figura 1.1). De igual modo, la evaluación de la calidad durante el uso permite la retroalimentación para mejorar un producto, y cuando se produce la evaluación permite la retroalimentación para mejorar un proceso. [6]

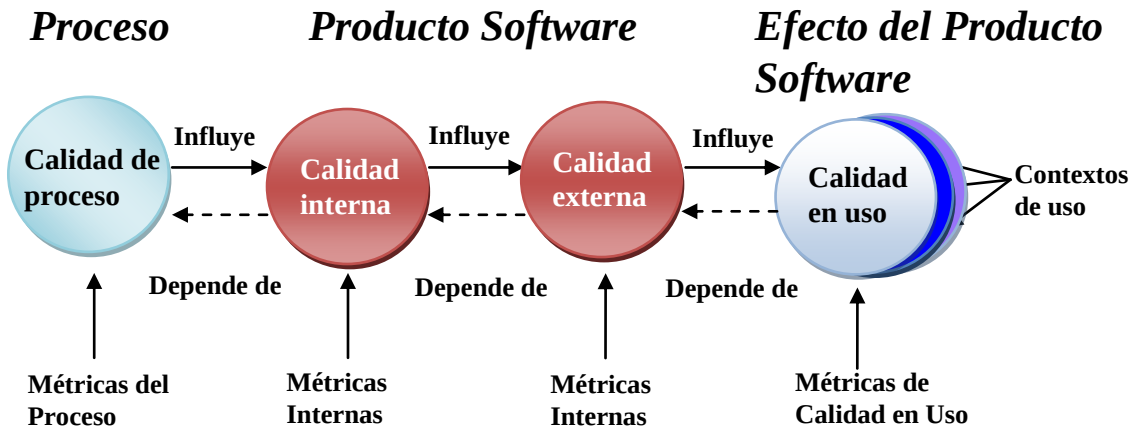



**Figura 1.1- Relación entre los diferentes enfoques hacia la calidad**

Como muestra la Figura 1.1, las métricas internas pueden ser aplicadas a los productos intermedios que se desarrollan a lo largo del ciclo de vida de desarrollo de un producto software, tales como solicitud de propuesta, especificación de requisitos, especificaciones de diseño o código fuente. Las métricas internas le proporcionan a los desarrolladores la habilidad de medir la calidad de estos productos intermedios, con lo cual se puede predecir la calidad del producto final. Esto le permite a los desarrolladores identificar los problemas que afecten la calidad e iniciar las acciones correctivas en las etapas tempranas del ciclo de vida de desarrollo del producto. [6]

Por su lado, las métricas externas pueden ser usadas para medir la calidad del producto software a través de la medición del comportamiento del sistema del cual el software forma parte (Figura 1.1). Las métricas externas solo pueden ser usadas durante las etapas de pruebas del proceso ciclo de vida y durante cualquier otra etapa operacional. Por último, las métricas de calidad en uso (Figura 1.1) miden si un producto resuelve las necesidades de usuarios específicos para alcanzar metas específicas con eficacia, productividad, seguridad y satisfacción en un contexto específico de uso. Esto solo puede lograrse en un entorno real del sistema. [6]

Las necesidades de calidad del usuario pueden ser especificadas como requisitos de calidad a través de las métricas de calidad en uso, las métricas externas y algunas veces las métricas internas. Estos requisitos especificados por las métricas deben ser usados como criterios cuando el producto se evalúa. [6]

Es recomendable usar métricas internas que tengan una relación lo más fuerte posible con los objetivos de las métricas externas, así ellas pueden ser usadas para predecir los valores de las métricas externas. Sin embargo, a menudo es difícil diseñar un modelo teórico riguroso que proporcione una relación fuerte entre las métricas internas y externas.

#### 1.2.3.2 ISO 25000:2005. SquaRE

SQuaRE (*Software Quality Requirements and Evaluation*) es una nueva serie de normas que se basa en ISO 9126 y en ISO 14598 (Evaluación del software). Uno de los principales objetivos de la serie SQuaRE es la coordinación y armonización del contenido de ISO 9126 y de ISO 15939:2002 (Measurement Information Model). ISO 15939 tiene un modelo de información que ayuda a determinar que se debe especificar durante la planificación, performance y evaluación de la medición. Para su aplicación, cuenta con los siguientes pasos: (1) Recopilar los datos, (2) Preparación de los datos y (3) Análisis de los datos.[9]

La integración de ISO 9126 e ISO 15939 permiten plantear un proceso de 4 pasos:[9]

1. Identificación de los requerimientos relacionados a la calidad del producto, es decir, seleccionar la parte del modelo de calidad (ISO/IEC 9126-n) que resulta relevante para la evaluación de calidad.

2. Identificación del contexto de interpretación. Es decir, selección de los valores de referencia y determinación de los target especificados en un contexto determinado.

3. Uso de las medidas derivadas de la etapa de preparación de los datos.

4. Comparación y análisis de los resultados obtenidos respecto de un conjunto de valores de referencia.

SQuaRE incluye un estándar de requerimientos de calidad. Está compuesto por 14 documentos agrupados en 5 tópicos: (1) Administración de la Calidad – 2500n, (2)

Modelo de Calidad – 2501n, (3) Medidas de Calidad – 2502n, (4) Requerimientos de Calidad – 2503n y (5) Evaluación de la Calidad – 2504n.

Estos 5 tópicos conforman la Arquitectura de SQuaRE.[9]

1. *Administración de la Calidad:* abarca:

    - Guía para SQuaRE – Overview de la estructura y terminología

    - Planificación y Administración – Provee una guía para planificar y administrar las evaluaciones del software.

2. *Modelo de Calidad:* describe el modelo de calidad interno - externo y la calidad en uso. Presenta características y sub-características.[9]

3. *Medidas de Calidad:* Medición de primitivas, Medidas para la calidad interna, Medidas para la calidad externa y Medidas para la calidad en uso.

4. *Requerimientos de Calidad*: permite habilitar la calidad del software a ser especificado en términos de requerimientos de calidad durante todo el ciclo de vida de un proyecto de software o adquisición, mantenimiento y operación.[9]

5. *Evaluación de la Calidad:* Evaluación de la Calidad, Proceso para desarrolladores, Proceso para compradores, Proceso para evaluadores y Documentación del módulo de evaluación.

Los beneficios de utilizar SQuaRE son: [9]

1. El modelo representa la calidad esperada del producto de software.

2. Planteo del desdoblamiento de las necesidades o expectativas en calidad en uso, calidad externa y calidad interna.

3. Permite una mayor eficacia en la definición del software.

4. Plantea la evaluación de productos intermedios.

5. Propone una calidad final a través de las evaluaciones intermedias.

6. Permite efectuar un rastreo entre las expectativas, requisitos y medidas de evaluación.

7. Mejora la calidad del producto.

#### 1.2.3.3 Modelo GQM

GQM (*Goal Question Metric*) define un objetivo, refina este objetivo en preguntas y define métricas que intentan dar información para responder a estas preguntas. Se puede aplicar a todo el ciclo de vida del producto, procesos, y recursos y se pude alinear fácilmente con el ambiente organizacional. Su principio básico es la medición, la cual debe ser realizada, siempre, orientada a un objetivo.[10]

*Pasos de GQM*

GQM lo podemos describir en términos de un proceso de seis pasos donde: [10]

Los tres primeros se basan en usar las metas de negocio para conducir a la identificación de las verdaderas métricas. Los últimos tres pasos se basan en recopilar los datos de las medidas y la fabricación del uso eficaz de las métricas para mejorar la toma de decisión.

1. Establecer las Metas

   - Desarrollar un conjunto de metas corporativas, de la división y del proyecto de negocio que estén asociadas a medidas de productividad y calidad.

2. Generación de Preguntas

   - Generar las preguntas que definen objetivos de la manera más completa y cuantificable posible.

3. Especificación de Medidas

   - Necesarias a ser recolectadas para contestar las preguntas y seguir la evolución del proceso y producto con respecto a las metas.

4. Preparar Recolección de datos

   - Desarrollar mecanismos para la recolección de datos.

5. Recolectar, Validar y Analizar los datos para la toma de decisiones.

   – Para proporcionar la realimentación de proyectos en una acción correctiva.

6. Analizar los datos para el logro de los objetivos y el aprendizaje.

   – Para determinar el grado de conformidad y hacer las recomendaciones para mejoras futuras.

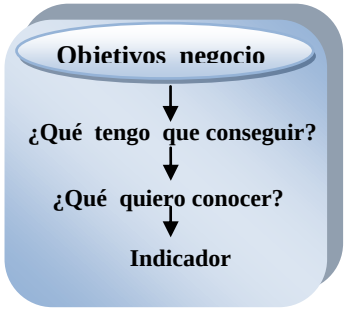



**Figura 1.3 - Pasos de GQM**

*Niveles de GQM*

1. Nivel Conceptual – Goals: Los objetivos identifican lo que queremos lograr respecto a los productos, procesos o recursos. [10]

   Objetos de la medición:

   – Productos: entregables y documentos que se producen durante el ciclo de vida de un sistema.

   – Procesos: actividades relacionadas con el software y asociadas generalmente al tiempo.

   – Recursos: elementos que los procesos utilizan para producir sus salidas.

2. Nivel Operacional – Questions: Las preguntas nos ayudan a comprender cómo satisfacer el objetivo. Abordan el contexto de la calidad desde un punto de vista particular. Para caracterizar el modo en que se va a realizar la valoración o para analizar el grado de cumplimiento de un objetivo específico. [10]

3. Las preguntas tratan de caracterizar al objeto de la medición con respecto a un aspecto de calidad concreto y tratan de determinar la calidad de dichos objetos desde el punto de vista seleccionado.

4. Nivel Cuantitativo – Metrics**:** Se asocia un conjunto de datos a cada pregunta, con el fin de proporcionar una respuesta de manera *cuantitativa*. [10]

Los datos pueden ser:

- *Objetivos*: si dependen únicamente del objeto que se está midiendo y no del punto de vista desde el que se captan (por ejemplo, el número de versiones de un documento).

- *Subjetivos*: si dependen tanto del objeto que se está midiendo como del punto de vista desde el que se captan (por ejemplo, el nivel de satisfacción del usuario).

Resultado: Seleccionar medidas existentes o definir nuevas medidas.

Para cada meta, puede haber varias preguntas y la misma pregunta se puede ligar a múltiples metas. Para cada pregunta puede haber múltiples métricas. Una métrica puede ser aplicable a más de una pregunta.

*Fases de GQM* [10]

1. Planificación

   - Se selecciona, define, caracteriza y planifica un proyecto para la aplicación de la medición, obteniéndose como resultado un plan de proyecto.

   - Esta fase por consiguiente toca los pasos 1 – 5.

2. Definición

   - Se define y documenta el programa de medición (objetivos, preguntas, métricas, hipótesis)

   - Esta comprende los tres primeros pasos

3. Recopilación de Datos

- Se recogen los datos reales de la medición
- Esta fase direcciona los pasos 4 y 5

4. Interpretación
   - Se procesan los datos recopilados para obtener respuestas a las preguntas definidas, a partir de las cuales se puede evaluar el logro de los objetivos planteados.
   - Esta fase implementa los pasos 5 y 6.

*Prácticas Clave de GQM* [10]

1. Tener a las personas adecuadas involucradas en el proceso de GQM.
2. Fijar objetivos de mediciones explícitos y especificarlos explícitamente.
3. No crear objetivos de mediciones falsos. No crear objetivos para lograr correspondencia con las métricas que ya tenemos.
4. Derivar métricas apropiadas.
5. Permanecer focalizado en los objetivos cuando se analizan datos.
6. Integrar las actividades de mediciones con las actividades regulares del proyecto.
7. Establecer la infraestructura necesaria para soportar el programa de mediciones.
8. Asegurar que las mediciones son vistas como una herramienta y no como el objetivo final.

GQM permite elegir métricas que se relacionan con las metas más importantes y problemas más urgentes. El proceso de interpretación de las medidas puede ser difícil cuando intervienen muchas métricas. Se deben considerar métricas útiles y relevantes desde el punto de vista de los objetivos del negocio. [10]

### 1.2.4 Proceso Unificado del Desarrollo de Software

La UCI para el buen desarrollo de los proyectos productivos utiliza el Proceso Unificado del Desarrollo de Software (RUP), lo cual es una metodología orientada a objeto que te ayuda a obtener un producto con calidad.

RUP integra técnicas de desarrollo así como elementos de metodologías anteriores; fue creada por Jacobson, Rumbaugh y Booch como resultado de varios años de trabajo y dedicación al mundo de la Ingeniería de Software. Define *quién, cómo, cuándo y qué* debe hacerse un proyecto y utiliza lenguajes de representación visual como es el UML[3].

Como filosofía RUP maneja 6 principios clave [11]:

1. Adaptación del proceso.

   El proceso deberá adaptarse a las características propias de la organización. El tamaño del mismo, así como las regulaciones que lo condicionen, influirán en su diseño específico. También se deberá tener en cuenta el alcance del proyecto.

2. Balancear prioridades.

   Los requerimientos de los diversos inversores pueden ser diferentes, contradictorios o disputarse recursos limitados.

3. Colaboración entre equipos.

   El desarrollo de software no lo hace una única persona sino múltiples equipos. Debe haber una comunicación fluida para coordinar requerimientos, desarrollo, evaluaciones, planes, resultados.

4. Demostrar valor iterativamente.

   Los proyectos se entregan, aunque sea de un modo interno, en etapas iteradas. En cada iteración se analiza la opinión de los inversores, la estabilidad y calidad del producto, y se refina la dirección del proyecto así como también los riesgos involucrados.

[3] UML: Lenguaje Unificado de Modelado

5. Elevar el nivel de abstracción.

Este principio dominante motiva el uso de conceptos reutilizables tales como patrón de software, Éstos se pueden acompañar por las representaciones visuales

6. Enfocarse en la calidad.

El control de calidad no debe realizarse al final de cada iteración, sino en todos los aspectos de la producción.

Las características principales del proceso son: [12]

- Guiado por los casos de uso: orientan el proyecto a la importancia para el usuario y lo que este quiere.

- Centrado en la arquitectura: relaciona la toma de decisiones que indican cómo tiene que ser construido el sistema y en qué orden.

- Iterativo e incremental: donde divide el proyecto en mini-proyectos donde los casos de uso y la arquitectura cumplen sus objetivos de manera más depurada.

RUP define como sus principales elementos: [12]

- Trabajadores (“quién”): Define el comportamiento y responsabilidades (rol) de un individuo, grupo de individuos, sistema automatizado o máquina, que trabajan en conjunto como un equipo. Ellos realizan las actividades y son propietarios de elementos.

- Actividades (“cómo”): Es una tarea que tiene un propósito claro, los procesos que se determinan en cada iteración, es realizada por un trabajador y manipula elementos.

- Artefactos (”qué”): Productos tangibles del proyecto que son producidos, modificados y usados por las actividades. Puede ser un documento, un modelo, elementos dentro del modelo, código fuente y ejecutables.

- Flujo de trabajo (“cuándo”): Secuencia de actividades realizadas por trabajadores y que produce un resultado de valor observable.

RUP implementa las siguientes mejores prácticas asociadas al proceso de Ingeniería de Software:[12]

- Desarrollo Iterativo.
- Manejo de los Requerimientos.
- Uso de una Arquitectura basada en componentes.
- Modelización Visual.
- Verificación Continua de la Calidad.
- Manejo de los Cambios.

El ciclo de vida de RUP: [12]

RUP divide el proceso en 4 fases, dentro de las cuales se realizan varias iteraciones en número variable según el proyecto y en las que se hace un mayor o menor hincapié en los distintas actividades.

En las iteraciones de cada fase se hacen diferentes esfuerzos en varias actividades:

**Inicio**: Se hace un plan de fases, se identifican los principales casos de uso y se identifican los riesgos. Se define el alcance del proyecto

**Elaboración**: se hace un plan de proyecto, se completan los casos de uso y se eliminan los riesgos

**Construcción**: se concentra en la elaboración de un producto totalmente operativo y eficiente y el manual de usuario

**Transición**: se Instala el producto en el cliente y se entrena a los usuarios. Como consecuencia de esto suelen surgir nuevos requisitos a ser analizados.

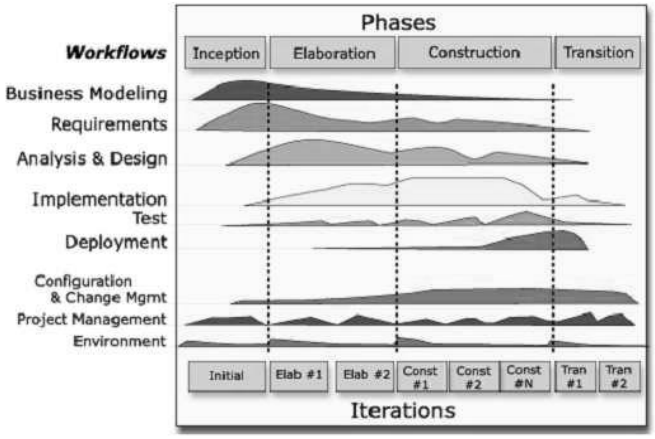



**Figura 1.4- Representa el ciclo de vida de RUP**

**FASE DE INICIO**[12]

Durante la fase de inicio las iteraciones hacen ponen mayor énfasis en los flujos de trabajo de modelamiento del negocio y de requisitos.

- *Modelamiento del negocio*

En esta fase el equipo se familiarizará más al funcionamiento de la empresa, sobre conocer sus procesos.

Las actividades implicadas incluyen:

1. Entender la estructura y la dinámica de la organización para la cual el sistema va ser desarrollado
2. Entender el problema actual en la organización objetivo e identificar potenciales mejoras.
3. Asegurar que clientes, usuarios finales y desarrolladores tengan un entendimiento común de la organización objetivo.

- *Requisitos*

En esta línea los requisitos son el contrato que se debe cumplir, de modo que los usuarios finales tienen que comprender y aceptar los requisitos que especifiquemos.

Las actividades implicadas incluyen:

1. Establecer y mantener un acuerdo entre clientes y otros stakeholders sobre lo que el sistema podría hacer.
2. Proveer a los desarrolladores un mejor entendimiento de los requisitos del sistema.

3. Definir el ámbito del sistema.
4. Proveer una base para estimar costos y tiempo de desarrollo del sistema.
5. Definir una interfaz de usuarios para el sistema, enfocada a las necesidades y metas del usuario.

**FASE DE ELABORACIÓN**[12]

En la fase de elaboración, las iteraciones se orientan al desarrollo de la línea base de la arquitectura, abarcan más los flujos de trabajo de requerimientos, modelo de negocios (refinamiento), análisis, diseño y una parte de implementación orientado a la línea base de la arquitectura.

- *Análisis y Diseño*

En esta actividad se especifican los requerimientos y se describen sobre como se van a implementar en los sistema

Las actividades implicadas incluyen:

1. Transformar los requisitos al diseño del sistema.
2. Desarrollar una arquitectura para el sistema.
3. Adaptar el diseño para que sea consistente con el entorno de implementación

**FASE DE CONSTRUCCIÓN**[12]

- *Implementación*

Se implementan las clases y objetos en ficheros fuente, binarios, ejecutables y demás. El resultado final es un sistema ejecutable.

Las actividades implicadas incluyen:

1. Planificar qué subsistemas deben ser implementados y en que orden deben ser integrados, formando el Plan de Integración.
2. Cada implementador decide en que orden implementa los elementos del subsistema. Si encuentra errores de diseño, los notifica.
3. Se integra el sistema siguiendo el plan.

- *Pruebas*

Este flujo de trabajo es el encargado de evaluar la calidad del producto que estamos desarrollando, pero no para aceptar o rechazar el producto al final del proceso de desarrollo, sino que debe ir integrado en todo el ciclo de vida.

Las actividades implicadas incluyen:

1. Encontrar y documentar defectos en la calidad del software.
2. Generalmente asesora sobre la calidad del software percibida.
3. Provee la validación de los supuestos realizados en el diseño y especificación de
4. requisitos por medio de demostraciones concretas.
5. Verificar las funciones del producto de software según lo diseñado.
6. Verificar que los requisitos tengan su apropiada implementación.

- *Despliegue*

Esta actividad tiene como objetivo producir con éxito distribuciones del producto y distribuirlo a los usuarios.

Las actividades implicadas incluyen:

1. Probar el producto en su entorno de ejecución final.
2. Empaquetar el software para su distribución.
3. Distribuir el software.
4. Instalar el software.
5. Proveer asistencia y ayuda a los usuarios.
6. Formar a los usuarios y al cuerpo de ventas.
7. Migrar el software existente o convertir bases de datos.

**FASE DE TRANSICIÓN** [12]

El objetivo de la fase de Transición es traspasar el software desarrollado a la comunidad de usuarios. Una vez instalado el software surgirán nuevos elementos que implicarán nuevos desarrollos (ciclos).

Incluye:

- Pruebas Beta para validar el producto con las expectativas del cliente
- Ejecución paralela con sistemas antiguos
- Conversión de datos
- Entrenamiento de usuarios
- Distribuir el producto

Objetivos

- Obtener autosuficiencia de parte de los usuarios.
- Concordancia en los logros del producto de parte de las personas involucradas.
- Lograr el consenso cuanto antes para liberar el producto al mercado.

## *1.3 Medición, Métrica e Indicador*

Una *medida* proporciona una indicación cuantitativa de la extensión, cantidad, dimensiones, capacidad o tamaño de algunos atributos de un proceso o producto. La *medición* es el acto de determinar una medida.[13]

Las medidas son las que se utilizan para comparar. En otras palabras podemos decir que:

- La medida captura una característica individual.
- La medición permite capturar dicha característica.
- La métrica permite relacionar y comparar mediciones.

En general, la medición persigue tres objetivos fundamentales[15]:

1. Entender qué ocurre durante el desarrollo y el mantenimiento.
2. Controlar qué es lo que ocurre en nuestros proyectos.
3. Mejorar nuestros procesos y nuestros productos.

*La métrica es* una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado[13]. Las métricas definen que es lo que se va a estimar y los métodos de estimación definen como una métrica es estimada. [13]

Un *indicador* es una métrica o combinación de métricas que proporcionan una visión profunda del proceso de desarrollo del software, del proyecto de software y del producto en si (esfuerzo, costo), por tanto las métricas son el fundamento de los indicadores.[14]

Un indicador proporciona una visión profunda que permite al gestor de proyectos o a los ingenieros de software ajustar el producto, el proyecto o el proceso.[13]

### 1.3.1 Clasificación de las medidas

Aunque hay muchas medidas de la calidad del software, la *corrección, facilidad de mantenimiento, integridad,* y *facilidad de uso* proporcionan indicadores útiles para el equipo del proyecto: [4]

**Corrección.** Un programa debe operar correctamente o proporcionara poco valor a sus usuarios. La corrección es el grado en el que el software lleva a cabo su función requerida.

**Facilidad de mantenimiento.** El mantenimiento del software cuenta con más esfuerzo que cualquier otra actividad de ingeniería del software. La facilidad de mantenimiento es la facilidad con la que se puede corregir un programa si se encuentra un error. No hay forma de medir directamente la facilidad de mantenimiento; por consiguiente, se deben utilizar medidas indirectas.

**Integridad.** En esta época de <hackers> y <firewalls>, la integridad del software ha llegado a tener mucha importancia. Este atributo mide la capacidad de un sistema para resistir ataques (tanto accidentales como intencionados) contra su seguridad. Para medir la integridad, se tienen que definir dos atributos adicionales: amenaza y seguridad. *Amenaza* es la probabilidad (que se puede estimar o deducir de la evidencia empírica) de que un ataque de un tipo determinado ocurra en un tiempo determinado. La *seguridad* es la probabilidad (que se puede estimar o deducir de la evidencia empírica) de que se pueda repeler el ataque de un tipo determinado.

**Facilidad de uso.** La facilidad de uso es un intento de cuantificar <lo amigable que puede ser con el usuario y se puede medir en función de cuatro características:

1. habilidad intelectual y/o física requerida para aprender el sistema.
2. el tiempo requerido para llegar a ser moderadamente eficiente en el uso del sistema.
3. aumento neto en productividad (sobre el enfoque que el sistema reemplaza) medida cuando alguien utiliza el sistema moderadamente y eficientemente.
4. valoración subjetiva (a veces obtenida mediante un cuestionario) de la disposición de los usuarios hacia el sistema.

Al medir se suele hacer la siguiente clasificación de los atributos [16]:

- Atributos internos: Son aquellos que se pueden medir en términos de la propia entidad (producto, proceso o recurso).Por ejemplo, es posible medir la longitud del código de un programa examinando exclusivamente la propia entidad
- .Atributos externos: Son los atributos que sólo se pueden medir con respecto al entorno de la entidad. De hecho resulta más importante el comportamiento de la entidad que la entidad en sí. Por ejemplo, la facilidad de uso de un programa

depende, no sólo del examen del programa, sino también del análisis de su ejecución, del usuario, de los trabajos ejecutados.

Se debe destacar que las mediciones del software pueden clasificarse en *medidas directas* y *medidas indirectas.* Entre las medidas directas del *proceso* de la ingeniería de software se incluyen el costo y el esfuerzo aplicados. Entre las medidas directas del *producto* se encuentran las líneas de código (LCD) producidas, velocidad de ejecución, y los defectos informados durante un período de tiempo establecido. Mientras que entre las medidas indirectas se incluyen la funcionalidad, calidad, complejidad, eficiencia, fiabilidad, facilidad de mantenimiento y otras "capacidades. [6]

### 1.3.2 Necesidad de las medidas

| Problema | Medir ayuda a |
|---|---|
| Correcciones | Proporcionar requerimientos verificables, expresados en términos medibles |
| Toma de decisiones | Proporcionar evidencia cuantificable para apoyar las decisiones |
| Falta de Control | Hacer más visible el desarrollo e identificar problemas anticipadamente |
| Exceso de gasto | Producir predicciones de coste y plazos justificables |
| Coste de Mantenimiento | Recomendar determinadas estrategias de pruebas e identificar los módulos problemáticos |
| Evaluación de nuevos métodos | Valorar los efectos en la productividad y calidad |

Tabla 1.1- Problemas más comunes en el desarrollo de un software [16]

### 1.3.3 Proceso de Medición

Las métricas, deben ayudar a la (1) la evaluación de los modelos de análisis y de diseño,(2) donde proporcionen una indicación de la complejidad de diseños procedimentales y de código fuente, y(3) ayuden en el diseño de pruebas más efectivas; [6] por lo que propone un proceso de medición que se caracteriza en cinco actividades.

La primera actividad es la *formulación*, donde se obtienen las medidas y métricas apropiadas para la representación del software en cuestión. Sigue la *colección*, como mecanismo empleado para acumular los datos necesarios para obtener las métricas formuladas. Luego se realiza un *análisis* del cálculo de las métricas y si se puede se aplican herramientas matemáticas. Después se hace una *interpretación*, evaluando los resultados de las métricas en un esfuerzo por lograr la calidad interna y se termina con una *retroalimentación* (feedback), donde las recomendaciones obtenidas de la

interpretación de las métricas se transmiten al equipo que trabaja en la construcción del software. [6]

## *1.4 Métricas*

Las métricas son un buen medio para entender, monitorizar, controlar, predecir y probar el desarrollo software y los proyectos de mantenimiento.

Una métrica es “una asignación de un valor a un atributo de una entidad de software, ya sea un producto o un proceso”.

John Wiley define métricas de calidad y criterios, donde cada métrica se obtiene a partir de combinaciones de los diferentes criterios. Las métricas proporcionan una indicación de la efectividad de las actividades de control y de la garantía de calidad en grupos o en particulares.

Las métricas tienen dos objetivos bien definidos, primero establecer un conjunto abierto de procedimientos e indicadores adecuados de calidad y segundo definir métodos para medir los indicadores de calidad del software, es este contexto es verdaderamente significativo que las características de calidad puedan ser cuantificables y medibles. El ciclo de vida del software es el conjunto de fases y actividades por las que atraviesa hasta que finaliza.

Tenerlo en cuenta es fundamental para el estudio de las métricas, ya que éstas pueden aplicarse a cualquier tarea, actividad o fase del producto o proceso, por ello es tan importante conocer a profundidad el proceso que se pretende evaluar a través de las métricas.

Lograr un software de buena calidad no solo requiere considerar algunas fases del ciclo de vida. En cada una de ellas existen contribuciones decisivas para la calidad. Un error en una de las fases iniciales puede ocasionar inmensos problemas en las posteriores lo que supone un considerable incremento de los costos a la vez que la probabilidad de un deterioro de la calidad o aumento del tiempo de salida del producto es muy alta[6].

Los principios fundamentales que deben seguir las métricas son[17]:

- Las métricas deben ser simples, objetivas, fáciles de coleccionar, fáciles de interpretar y difíciles de malinterpretar.
- La colección de las métricas debe ser automática o sea, no interferir en las actividades de los desarrolladores.

- Las métricas deben contribuir a la evaluación de la calidad temprana en el ciclo de vida, cuando los esfuerzos por mejorar la calidad del software son efectivos.
- Los valores absolutos y las tendencias de las métricas, deben ser usados activamente por el personal administrativo y el personal de ingeniería, para comunicar progreso y calidad en un formato coherente.
- La selección de un mínimo o más extensivo conjunto de métricas, dependerá de las características y contexto del proyecto: Si es muy grande o si tiene restricciones de seguridad o de confiabilidad de los requerimientos; y si el equipo de desarrollo y de valoración (evaluación) es conocedor de las métricas, lo cual hará muy útil coleccionar y analizar las métricas técnicas.

*Relaciones entre métricas* [15]

- Una métrica está definida para uno o más *atributos.*
- Dos métricas pueden relacionarse mediante una función de transformación.
- El tipo de dicha función de transformación va a depender del tipo de *escala* de ambas métricas.
- Una métrica puede expresarse en una *unidad* (sólo para métricas cuya escala sea de tipo intervalo o ratio)

Las instituciones que trabajan con proyectos de software están sometidas a las presiones y desafíos del mercado por lo que la gestión de proyectos y mejora de la calidad son actividades fundamentales para su supervivencia. A pesar de que la Ingeniería de Software ha introducido y popularizado una serie de estándares para medir y certificar la calidad, tanto del sistema a desarrollar, como del proceso de desarrollo en sí, y además ha surgido un número creciente de herramientas automatizadas para ayudar a definir y aplicar un proceso de desarrollo de software efectivo se agudiza una alta incidencia de fallos en los proyectos de software. Por esto se ve parte de la solución en la utilización de métricas durante todo el ciclo de desarrollo del software enfatizando en las primeras etapas a las que menos se le ha dedicado en lo que a ello respecta. [18]

Existen cuatro razones fundamentales por las que es necesario realizar mediciones, tanto al proceso como al producto software y sus recursos.

Estas son:

1. Caracterizar, para comprender mejor los procesos, productos, recursos y entornos, además para establecer las líneas base para comparaciones con evaluaciones futuras
2. Evaluar, para conocer cuándo los proyectos y procesos andan por el camino equivocado, de modo que se puedan poner bajo control, además para valorar la consecución de los objetivos de calidad y evaluar las mejoras del proceso en los productos
3. Predecir, para hacer planificaciones, que significa aumentar la comprensión de las relaciones de los procesos, productos y construcción de modelos de estas relaciones, así los valores que se observan para algunos atributos pueden utilizarse para predecir otros, además las medidas de predicción son la base para la estimación de costos, tiempo, calidad, análisis de riesgos y realización de cambios
4. Mejorar, la calidad del producto y rendimiento del proceso cuando se ha recogido información cuantitativa que ayuda a identificar obstáculos, problemas de raíz e ineficiencias. [18]

### 1.4.1 Clasificación de las métricas

| **Clasificación de las Métricas** | | |
|---|---|---|
| **Entidades** | **Atributo Interno** | **Atributo Externo** |
| ***Productos*** | | |
| **Especificaciones** | Tamaño, reutilización, modularidad, redundancia, funcionalidad, sintaxis, corrección | Fácil de comprender, fácil mantenimiento |
| **Diseño** | Tamaño, reutilización, modularidad, cohesión, acoplamiento | Calidad, complejidad fácil mantenimiento |
| **Código** | Tamaño, reutilización, modularidad, funcionalidad, cohesión, complejidad algorítmica, acoplamiento | Calidad, complejidad fácil mantenimiento, fiabilidad de uso |
| | Tamaño Cobertura | Calidad |

| | | |
|---|---|---|
| ***Procesos*** | | |
| | Esfuerzo, número de cambios | Calidad, coste, estabilidad |
| | Esfuerzo, número de errores en las especificaciones | Efectividad-costes |
| | Esfuerzo, número de errores encontrados | Efectividad-costes, fiabilidad, estabilidad |
| ***Recursos*** | | |
| | Edad, coste | Productividad, Experiencia |
| | Tamaño, nivel de comunicación | Productividad, Calidad |
| | Precio, tamaño | Usabilidad, fiabilidad |
| | Precio, memoria, rapidez | Fiabilidad |

Tabla 1.2- Clasificación de las métricas **[16].**

### 1.4.2 Métricas de software

Se define métricas de software 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 estas técnicas mejorará el proceso y sus productos". [17]

Las métricas del software son de gran importancia ya que ayudan a:

- Seguir objetivamente a los proyectos, teniendo en cuenta impactos en el cumplimiento de plazos, costos y calidad (satisfacción del cliente).
- Estimar y predecir tendencias en función de la organización, los procesos, la gestión del cambio, la gestión de la configuración, medidas de software, etc.
- Evaluar de la productividad.
- Tomar decisiones objetivas.
- Comunicarse a través de un lenguaje común.
- Identificar los puntos críticos del proyecto para garantizarlos.

- Gestionar los riesgos.
- Identificar las mejores prácticas por comparación objetiva de métricas.
- Identificar y eliminar lo antes posible las causas de los principales defectos.
- Mejorar el entendimiento del proceso.
- Mejorar el proceso software mediante una estrategia a medio y largo plazo.

Se utilizan fundamentalmente para [14]:

- Obtener las bases para la estimación.
- Seguir el progreso de los proyectos.
- Determinar la complejidad (relativa).
- Ayudar a comprender cuando se ha alcanzado un estado deseado de calidad.
- Analizar los defectos.
- Validar experimentalmente las mejores prácticas.

Las métricas del software responden a dos objetivos fundamentales, la valoración y la estimación. Las principales magnitudes dentro de la valoración son la calidad, fiabilidad y la productividad; mientras que las magnitudes de la estimación corresponden al esfuerzo y al tiempo.

Algunas propiedades que deben acompañar a las métricas para que sean efectivas [14]:

- Simple y fácil de calcular: debería ser relativamente fácil de aprender a obtener la métrica y su cálculo no obligará a un esfuerzo o a una cantidad de tiempo inusuales.
- Empírica e intuitivamente persuasiva: la métrica debería satisfacer las nociones intuitivas del ingeniero de software sobre el atributo del producto en cuestión (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 deberían depender de los caprichos de la sintaxis o semántica del lenguaje de programación.
- Un mecanismo eficaz para la retroalimentación de la calidad: la métrica debería suministrar al desarrollador de software información que le lleve a un producto final de superior calidad.

Las métricas de software deben medir el proceso, el proyecto, el producto y los recursos, partiendo del hecho de que[17]:

- proceso: es la secuencia o las actividades invocadas para producir el producto de software (y otros artefactos).
- producto: son los artefactos del proceso, incluyendo el software, los documentos y modelos.
- proyecto: son todos los recursos del proyecto, actividades y artefactos.
- recursos: son las personas, los métodos y herramientas, el tiempo, esfuerzo y presupuesto, disponibles para el proyecto.

#### 1.4.2.1 Métricas de producto

Los productos son artefactos que pueden ser documentos, componentes, modelos, diagramas, módulos, a los cuales se les aplican métricas para obtener mediciones de cada uno de estos productos. Las métricas del producto describen características como el tamaño, complejidad, rasgos del diseño, rendimiento y nivel de calidad [14]

En general las características a medir de los artefactos del producto son [14]

1. Tamaño: Las métricas del tamaño del producto se refieren generalmente al volumen del producto desarrollado. Incluyen líneas de código (LOC), número de ficheros, páginas de la documentación.
2. Calidad:
    - Defectos: indicadores de que un artefacto no funciona como ha sido especificado, o cualquier otra característica indeseable. Complejidad de una estructura o un algoritmo: mientras mayor sea la complejidad y más difícil sea de comprender y modificar la estructura del sistema, mayor probabilidad habrá de que falle.
    - Acoplamiento: mediciones de cuántos elementos del sistema están interconectados y cuán extensivamente.

- Cohesión: mediciones de cuán bien un elemento o un componente cumple con los requerimientos de tener un sólo y bien definido propósito.
- Primitividad: el grado en el cual las operaciones o métodos de una clase pueden estar compuestos por otros de la misma clase.

3. Totalidad: medición de la magnitud en la cual un artefacto cumple con todos los requerimientos (plan / real).
4. Rastreabilidad: Indicadores de que los requerimientos de determinado nivel se están satisfaciendo por determinados artefactos, o que todos los artefactos tengan razón de existir.
5. Volatilidad: el grado de cambio de un artefacto debido a defectos o a cambios en los requerimientos.
6. Esfuerzo: medición del trabajo (Unidad de tiempo del personal) que es requerido para producir un artefacto.

#### 1.4.2.2 Métricas de proyecto

El proyecto debe ser caracterizado en términos de tamaño, tipo, complejidad y formalidad porque condicionan expectativas sobre las distintas tendencias a seguir a la hora de las mediciones. Algunas de las características que describen son: el número de programadores de un software, el costo, la planificación, la productividad del equipo, entre otros. [14]

Las principales métricas a tener en cuenta son [14]:

- 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.

Durante el desarrollo de un proyecto surgen varios indicadores, que permiten [14]:

- Evaluar el estado del proyecto en curso.
- Seguir la pista de riesgos potenciales.

- Detectar áreas problemáticas antes de que se conviertan en críticas.
- Ajustar el flujo y las áreas de trabajo.
- Evaluar la habilidad del equipo de proyecto en controlar la calidad de los productos de trabajo de la Ingeniería de Software.

La utilización fundamental de las métricas del proyecto es para minimizar la planificación de desarrollo, guiando los ajustes necesarios que eviten retrasos y mitiguen problemas y riesgos potenciales; así como para evaluar la calidad de los productos en el momento actual modificando el enfoque técnico para mejorar la calidad. [14]

#### 1.4.2.3 Métricas de proceso

Los procesos de software pueden definirse como los pasos definidos para determinar quién, cuándo, cómo y dónde, debe hacer cada actividad dentro del proceso de desarrollo de software. Las métricas del proceso brindan un mayor enfoque sobre la calidad lograda como consecuencia de un proceso repetible y ordenado. Este proceso es un factor clave y controlable para mejorar la calidad del software y el rendimiento del trabajo. [14] Es importante conocer que las métricas del proceso se van a obtener de las métricas del proyecto.

Las métricas del proceso dependen esencialmente del entorno de desarrollo. Es necesario medir atributos específicos de los propios procesos, como el tiempo empleado, su coste y el esfuerzo requerido. La relación entre las medidas de los resultados obtenidos en un proceso y los recursos usados en él permitirá medir la productividad, atributo este clave para estimar costo y esfuerzo[14].

Las métricas del proceso se coleccionan de todos los proyectos y durante un extenso período de tiempo con el objetivo de proporcionar indicadores que lleven a mejoras de los procesos de software a largo plazo. En algunos casos, se pueden utilizar las mismas métricas del software para determinar tanto el proyecto como los indicadores del proceso[14].

Los indicadores utilizados dentro del proceso permiten:

- Al gestor evaluar lo que funciona y lo que no.
- A la organización, tener una visión profunda de la eficacia de un proceso ya existente.

Es importante destacar que el proceso es el único factor de los controlables al mejorar la calidad del software y su rendimiento como organización.

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

Las métricas del proceso del software pueden proporcionar beneficios significativos a medida que una organización trabaja por elevar su nivel global de madurez del proceso. A medida que una organización está más conforme con la recopilación y utiliza métricas de proceso, la derivación de indicadores simples abre el camino hacia un enfoque más estricto llamado *Mejora Estadística de Proceso del Software (MEPS).* [6]

#### 1.4.2.5 Métricas de Calidad en el Software

Para producir sistemas, aplicaciones o productos de alta calidad los ingenieros del software deben aplicar métodos efectivos junto con herramientas modernas dentro del contexto de un proceso maduro de desarrollo de software, esto incluye medir si la alta calidad se está llevando a cabo.[6]

La calidad de un sistema, aplicación o producto es tan buena como los requisitos que describen el problema, el diseño que modela la solución, el código que conduce a un programa ejecutable y las pruebas que ejercitan el software para detectar errores, en este sentido los desarrolladores deben realizar y utilizar mediciones.

Para alcanzar esta evaluación de la calidad en tiempo real, se deben utilizar medidas técnicas que evalúan la calidad con objetividad, no con subjetividad.

A pesar de que se pueden recolectar varias medidas de calidad, el primero de los objetivos en el proyecto es medir los errores y defectos.

Las métricas que provienen de estas medidas proporcionan una indicación de la efectividad de las actividades de control y de aseguramiento de la calidad en grupos o en particulares.[6]

#### 1.4.2.6 Métricas de Productividad

La productividad se va a definir como la razón entre entradas/salidas. Para el caso de la ingeniería de software, se define como la cantidad de esfuerzo requerido para lograr un cierto grado de productividad [14].

Las métricas orientadas a la productividad se centran en el rendimiento del proceso de la ingeniería de software, o sea, en que tan productivo es el software que se va a diseñar. Cuando se trata de establecer métricas de productividad y calidad en la construcción de software, o para realizar estimaciones de coste o duración, es imprescindible disponer de una medida fiable y comprensible del tamaño de lo que se construye. Por este motivo cambios bruscos en las medidas de productividad entre proyectos, es una indicación de que no se está siguiendo un proceso estándar. En la medida de que los grupos de trabajo se consoliden en un proceso estándar de desarrollo, los rangos de productividad se deben estabilizar y ser más consientes. [14]

En la actualidad, la industria del software necesita tener un control y seguimiento más preciso, predecible y repetido sobre los procesos de producción y los productos de software. Con el objetivo de planear el progreso de los proyectos, mediante las métricas se obtienen las bases para estimar, ayudando a analizar la complejidad y obtener mejores prácticas con vistas a lograr mejores decisiones.

## *1.5 Evaluación de la Calidad en la UCI*

La Universidad de las Ciencias Informáticas es un centro que se caracteriza por tener un amplio desarrollo en la industria del software; son innumerables los proyectos productivos que se han estado realizando en la universidad esto ha traído como consecuencia que cada día se incrementen las investigaciones sobre la calidad de estos proyectos.

En la amplia gama de investigaciones acerca de este tema la UCI ha tenido la necesidad de evaluar el desarrollo de los proyectos de manera cuantitativa. Al realizar un conversatorio con la Ing. Dayami Rodríguez miembro del Departamento de Calidad UCI nos comenta que se han dado a la tarea de investigar y realizar una propuesta de métricas de seguimiento y control de proyectos productivos, la cual se le presentara a la dirección de la universidad.

La propuesta se divide en dos líneas bases:

1. Métodos de estimación
    - Se ha estado realizando entrevistas en los proyectos productivos para determinar cuales son las métricas básicas (tiempo, productividad) y los métodos de estimación (COCOMO) que se ajustan a las necesidades del proyecto.
2. Procedimiento de medición

- Se ha estado realizando un levantamiento de información por los proyectos productivos, como por ejemplo: realizar registro histórico del proyecto, definir bien sus características y clasificarlo en base a ellas.
- Se ha entrevistado a los jefes de proyectos para determinar las medidas, métricas o indicadores que se ajustan sus proyectos productivos.
- Procedimiento inicial con las métricas más importantes y aplicarlas a otros productos semejantes.

Nuestra investigación dentro de esta propuesta se va a centrar principalmente en los procedimientos de medición y en especial en las métricas que puedan ayudar a mejorar la calidad en el flujo de trabajo de análisis y diseño de los proyectos productivos de la UCI.

## *1.6 Conclusiones del capítulo*

En este capítulo se fundamentaron diferentes temas que son importantes para el buen entendimiento de la investigación. Se abordó el tema de la calidad del software y del proceso dentro de su creación, su definición, control y evaluación. Se investigó sobre algunos de los estándares de calidad que están muy relacionados no solo con las métricas sino con el proceso de medición a tener en cuenta a la hora de aplicarlas, la mayoría de estos estándares responden al producto de software por esta razón para elaborar la propuesta se debe consultar la ISO 9126 parte 3 – Métricas Internas y el libro de Preesman, que refieren algunas métricas que se pueden adaptar al proceso de Análisis y Diseño de un software. Como metodología de planificación y desarrollo se estudió RUP puesto que es la metodología más utilizada en el desarrollo del software de la universidad. Se conceptualizó medición, métrica e indicador ya que sirven de base para poder explicar el proceso de medición que utilizaremos para plantear la propuesta de solución. Al realizar un conversatorio con la Ing. Dayami Rodríguez miembro del Departamento de Calidad UCI se pudo abordar sobre el trabajo que se está realizando en relación con las métricas en la universidad.

# Capítulo 2
# Propuesta de Solución

## *2.1 Introducción*

En el presente capítulo para un mejor entendimiento e interpretación de las métricas de los líderes de proyecto se describen detalladamente los flujos de trabajo de análisis y diseño, como lo propone la metodología RUP. Se seleccionan las medidas y se realiza la propuesta de métrica del flujo de trabajo, posteriormente identificamos y realizamos la interpretación de los indicadores derivados de las métricas.

## *2.2 Flujo de trabajo de Análisis y Diseño*

El flujo de trabajo análisis y diseño en nuestra investigación lo estudiamos de forma unida, como se plantea en la metodología RUP. Tiene como objetivo transformar los requisitos el diseño del futuro sistema, por lo que las actividades del análisis proporcionan una correcta y fácil entrada al diseño, ya que una vez comprendido los requisitos se aumenta el nivel de especificidad de los mismos y se garantiza el cubrimiento de los requisitos funcionales y no funcionales.

Tanto en el análisis como en el diseño se crean diferentes diagramas, como por ejemplo los diagramas de interacción (secuencia y colaboración) los cuales son dos de los cinco tipos de diagramas de UML que se utilizan para modelar los aspectos dinámicos de los sistemas.

### 2.2.1 Análisis

La UCI para el buen desarrollo del software orientado a objeto utiliza la metodología RUP, la cual se describió en el epígrafe 1.2.4. Esta metodología está compuesta por diferentes flujos de trabajo y el análisis es uno de ellos.

El análisis es la transformación disciplinada de los requerimientos de información de un sistema o área, en una especificación funcional, expresada en términos lógicos y usando metodologías estándares[1], por lo que se interesa solamente por los requisitos funcionales. Es el proceso de determinar QUE se necesita hacer, antes de

decidir COMO debe hacerse. Utiliza una combinación de texto y de diagramas, para representar los requisitos de datos, funciones y comportamientos, que es relativamente fácil de entender y, más importante aún, sencillo para revisar su corrección, completitud y consistencia.

Responde a las preguntas: ¿Qué es lo que hace el sistema? Y ¿Qué hará el nuevo sistema?.

Las actividades del análisis se adentran en el problema a resolver, representando una vista interna del sistema y que utilizando el lenguaje de los desarrolladores se refinan los requisitos y se estructuran en base de clase y paquetes, por lo que el propósito del análisis es: [19]

- Conseguir una comprensión más precisa de los requisitos, refinarlos y estructurarlos.

- Utilizar el lenguaje de los desarrolladores para analizar con profundidad los requisitos funcionales.

- Proporcionar una visión general del sistema.

Los objetivos principales del análisis son[1]:

- Obtener el conocimiento detallado del sistema de información actual y de todos los requerimientos y necesidades de información adicionales.

- Realizar una reevaluación del pre-análisis, con base en un conocimiento más detallado del sistema, que permita hacer estimativos más reales, llegando a determinar factibilidades, alternativas y cronogramas más precisos, además de confirmar o aclarar completamente el alcance planteado anteriormente.

- Servir de puente de comunicación entre la parte administrativa y la parte técnica. Se traduce el lenguaje del usuario a un lenguaje de sistemas, que permita su desarrollo posterior.

- Emitir una formulación específica de los modelos lógicos, que representen lo que va a ser el nuevo sistema, con base en los requerimientos planteados por el usuario.

- Plantear consideraciones para el desarrollo del resto de etapas, especialmente para el diseño.

- Encontrar lo que la organización requiere que se haga antes de comenzar a imaginarse cómo hacerlo.

Las principales actividades y artefactos que se construyen en el análisis se muestran en el Anexo No 1.

El análisis es un flujo de trabajo que no es obligatorio llevarlo a cabo pero es un paso de transición entre el levantamiento de requisito y el diseño que son dos flujos de extrema importancia en la confección de proyectos. En este flujo se establecerá el fundamento para el diseño.

La especificación de requisitos del software se produce en la culminación de la tarea de análisis, esto puede parecer una tarea relativamente sencilla, pero el contenido de comunicación es muy denso, por lo que abundan las ocasiones para las malas interpretaciones o falta de información. Es muy probable que haya ambigüedades. En este flujo de trabajo es de suma importancia realizar una correcta especificación de requisitos, pues se traduce del lenguaje del usuario al lenguaje del sistema.

#### 2.2.1.1 Pasos en el desarrollo del análisis

Los siguientes son los pasos a seguir para lograr un desarrollo coherente y serio en el análisis de un sistema de información. Cada una de estas tareas debe estar claramente documentada, en el manual de factibilidad del sistema. [1]

1. Análisis del sistema actual

Se busca conocer con todo el detalle, cuál es el funcionamiento actual del área y en especial, cuál es el manejo y flujo de la información producida y que llega al área. Es una labor totalmente descriptiva y de conocimiento por parte del grupo. Incluye:[1]

- Definición de objetivos y funcionamiento del área.

- Información que alimenta al sistema.

- Definición de los procesos de información existentes, qué hacen, cómo lo hacen, con qué información lo hacen y qué resultados arrojan.

Esto se logra a través de diferentes medios:

- Manuales de normas administrativas y operativas.
- Manuales de procedimientos administrativos y operativos.
- Diagramas de flujo de documentos.
- Entrevistas con usuarios.
- Observación directa de los procesos y manejo de la información.

2. Análisis requerimientos del nuevo sistema

Se pretende conocer con todo el detalle, cuáles son los requerimientos de información del usuario. El usuario presenta inquietudes que él mismo NO sabe si pueden ser resueltas y omite otras que considera secundarias. Comprende:[1]

- Conocimiento de la lista de necesidades y nuevos requerimientos de información del usuario (al detalle).
- Exploración de otros posibles requerimientos del sistema, teniendo en cuenta las posibilidades técnicas de cómputo y las necesidades futuras del área.
- Clasificación y agrupación de los requerimientos.

Normalmente esta información se obtiene a través de entrevistas con el usuario y de observación directa de los procesos.

3. Revaluación del pre-análisis

Luego de conocer en detalle el funcionamiento del sistema actual y la definición concreta de los requerimientos del nuevo sistema, se debe hacer una reevaluación de los puntos tratados en el estudio de factibilidad, debido a que en la primera etapa no se conocía en forma detallada el sistema. Puede incluir:[1]

- Modificación de la definición del sistema.
- Modificación de los recursos asignados.

- Modificación de los estimativos.
- Inclusión de nuevas alternativas de desarrollo.
- Cambios en la factibilidad del sistema.
- Modificación al cronograma.

4. Desarrollo de las especificaciones del sistema propuesto.

Es el trabajo central del análisis, donde se representan los requerimientos del sistema (nuevo y actual), en un modelo lógico y estructurado que dé respuesta a las verdaderas necesidades de información del área. Se requiere tener claramente definido el sistema actual, los nuevos requerimientos y el estudio de factibilidad revaluado.[1]

Modelo del análisis

En la construcción del modelo de análisis se tienen que identificar las clases que describen la realización de los casos de uso, los atributos y las relaciones entre ellas. Con esta información se construye el diagrama de clases del análisis, que por lo general se descompone para agrupar las clases en paquetes. Esta descomposición tiene impacto por lo general en el diseño e implementación de la solución. [20]

Identificación de clases del análisis:

Las clases que se identifican están asociadas con el contexto del dominio del problema por lo que representan conceptos y relaciones. [20]

- **Clase interfaz:** En este tipo de clase se modela la interacción entre el sistema y sus actores.
- **Clase entidad:** Estas clases modelan información que posee una larga vida y que a menudo es persistente y fenómenos, conceptos y sucesos que ocurren en el mundo real. La fuente principal de obtención son las clases entidades del negocio y el glosario de términos que se ha ido elaborando. Algunos autores proponen un estudio del texto, a partir de las frases nominales, de manera que los sustantivos representan objetos y clases.

- **Clase de control:** Las clases de control coordinan el trabajo de uno o unos pocos casos de uso, coordinando las actividades de los objetos que implementan la funcionalidad del caso de uso, por lo que definen el flujo de control y las transacciones dentro de un caso de uso delegando el trabajo a otros objetos.

Diagrama de clases del análisis: [20]

Un diagrama de clases del análisis es un artefacto en el que se representan los conceptos en un dominio del problema.

Organización en paquetes:

Los paquetes son un mecanismo de organización de elementos que subdividen el modelo en otros más pequeños que colaboran entre sí. Tienen que ser: [20]

- Cohesivos: sus contenidos deben estar fuertemente relacionados.
- Débilmente acoplados: minimizar las dependencias entre paquetes.

Este particionamiento debe hacerse sobre la base de los requerimientos funcionales y el dominio del problema; y debe ser reconocible por las personas con conocimiento del dominio. Para ello se propone asignar la mayor parte de un cierto número de casos de uso a un paquete concreto. Se pueden seguir los siguientes criterios de agrupamiento:

- Casos de uso requeridos para dar soporte a un determinado proceso de negocio.
- Casos de uso requeridos para dar soporte a un determinado actor del sistema.
- Casos de uso que están relacionados mediante relaciones de generalización y extensión.
- Casos de uso que se ejecutan en un nodo.

Hay un tipo de paquete que se conoce como paquete de servicio ya que contiene un conjunto de clases relacionadas funcionalmente y un conjunto coherente de acciones relacionadas funcionalmente, que se utilizan en varios casos de uso. Son altamente reutilizables, por lo que no se cumple con ellos el principio de bajo acoplamiento, y pueden emplearse en varias realizaciones de casos de uso diferentes.[20]

En el análisis se pueden usar diferentes criterios de agrupamiento de paquetes, pero mediante los paquetes de análisis se pueden definir subsistemas de diseño, por esto se deben agrupar teniendo en cuenta el mismo criterio y así se va perfilando el sistema.

### 2.2.2 Diseño

El diseño es de extrema importancia en el ciclo de vida de RUP pues contribuye a la creación de una arquitectura estable y sólida y a la vez fomentar las bases en el modelo de implementación, por esto el diseño y la implementación de un sistema están muy relacionados.

Importancia del diseño: [1]

- Organiza las ideas referentes al desarrollo de un nuevo sistema, facilitando el
- trabajo por realizar en la etapa de construcción.
- Define claramente los componentes que tendrá el nuevo sistema, a nivel de bases de datos, procesos e interfaces.
- "Descubre" la estructura física que tendrá el nuevo sistema.
- Toma en cuenta el diseño de formatos tanto de entrada de datos, como de salidas del propio sistema.
- Proporciona una visión inicial para los usuarios, de cómo será su interacción con el sistema, a través de los prototipos.
- Da claridad para definir la arquitectura necesaria que soportará al nuevo sistema.

En el diseño modelamos el sistema y encontramos su forma (incluida la arquitectura) para que soporte todos los requisitos, incluyendo los no funcionales y las restricciones que se le suponen. La entrada del modelo de análisis en el diseño proporciona una correcta comprensión de los requisitos.

Concretamente podemos definir como propósitos del diseño: [19]

- Transformar los requerimientos en un diseño de como el sistema debe ser.

- Desarrollar una robusta arquitectura del sistema.

- Adaptar el diseño para que se corresponda con el entorno de implementación, diseñando sus funcionalidades.

Al realizar la actividad del diseño dentro de la metodología RUP se obtiene como salida del flujo, las cuales son e inicio de la implementación del programa:

Arquitectura del sistema

Anteriormente nos referimos que en el diseño es donde se termina de conformar la arquitectura del sistema, es decir se obtiene una arquitectura sólida y concreta del sistema pero, ¿qué se entiende por arquitectura de software?

Según[2] la arquitectura de software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución.

Modelo de Diseño

El modelo de diseño es un modelo de objetos que describe la realización de los casos de uso, y sirve como una abstracción del modelo de implementación y el código fuente. Es usado como una entrada inicial en las actividades de implementación y prueba. Es una abstracción de la implementación del sistema. Es usado para concebir un documento del diseño del sistema de software. Es abarcador, compuesto por artefactos que engloban todas las clases del diseño, subsistemas, paquetes, colaboraciones, y las relaciones entre ellos. [19]Dentro del modelo del diseño tenemos:

Clases del diseño

Una clase de diseño es una construcción similar en la implementación del sistema: [19]

- El lenguaje utilizado para especificar una clase del diseño es lo mismo que el lenguaje de programación. Las operaciones, atributos, tipos, visibilidad (*public, protected, private*) se pueden especifican con la sintaxis del lenguaje elegido.

- Las relaciones entre clases de diseño se traducen de manera directa al lenguaje.

- Los métodos de una clase del diseño tienen correspondencia directa con el correspondiente método en la implementación de las clases.

- Se pueden postergar algunos requisitos a implementación (por ejemplo: manera de nombrar los atributos, operaciones).

Una clase de diseño puede proporcionar interfaces si tiene sentido hacerlo en el lenguaje de programación.

Los diagramas de clases son diagramas de estructura estática que muestran las clases del sistema y sus interrelaciones. Los diagramas de clase son el pilar básico del modelado con UML, siendo utilizados tanto para mostrar lo que el sistema puede hacer (análisis), como para mostrar cómo puede ser construido (diseño).

Subsistema de diseño

El diseño se estructura en subsistemas, el cual está conformado por otros elementos del modelo. Desde el exterior un subsistema es un único elemento del diseño que colabora con otros elementos del modelo para cumplir sus responsabilidades Internamente un subsistema es una colección de elementos del modelo (clases del diseño y otros subsistemas) que entienden la interfaz y el comportamiento de la especificación del subsistema. [19]

Paquetes de Diseño

Un paquete de diseño es una colección de clases relaciones, realizaciones de casos de uso, diagramas y otros paquetes; es usado para estructurar el modelo de diseño mediante su división en partes más pequeñas. Tiene como propósito agrupar elementos del modelo de diseño relacionados con propósitos organizacionales y frecuentemente para administración de configuración. Diferente al artefacto subsistema del diseño, un paquete de diseño no ofrece ninguna interfaz formal aunque debe exponer algo de su contenido (marcado como público) que ofrece comportamiento. [19]

#### 2.2.2.1 Pasos en el desarrollo del diseño.

Para el correcto desarrollo en el diseño, es necesario seguir varios pasos que te ayudan a organizar el trabajo en este flujo, cada una de estos pasos es necesario documentarlos.

Según [1] los pasos en el desarrollo del diseño son:

1. Definir la estructura del sistema (diseño global).

Su objetivo es presentar el sistema bajo una estructura funcional que coordine y oriente la ejecución de todos los módulos que lo componen. Se construye a partir del diagrama de flujo de datos y el Diccionario de datos. Se visiona la composición del sistema desde el punto de vista de componentes de desarrollo para la etapa de construcción. Esta estructura se evalúa con base en técnicas de diseño estructurado como cohesión y acople, que se describirán más adelante. La tarea es recursiva, es decir, se va modificando y decantando la estructura, en la medida que se vaya evaluando con el usuario.

2. Descomposición funcional de módulos.

Esta técnica es la empleada para elaborar la estructura del sistema. Consiste en descomponer en forma sucesiva un módulo en otros módulos de más baja jerarquía, hasta lograr el detalle suficiente en la asignación de funciones a estos.

Reglas para la descomposición[1]:

- Cualquier estructura tendrá un módulo general o coordinador.
- Cada módulo, si lo requiere, se subdividirá en otros.
- Cada módulo subordinado, será coordinado por sus padres (un módulo puede tener varios padres).
- Deben existir por lo menos dos módulos al mismo nivel de descomposición.

3. Diseño de módulos. (Diseño detallado)

Es la descripción y representación detallada de cómo cada módulo de la estructura definida, ejecutará su trabajo con el fin de facilitar el proceso de construcción.

Es básico en este punto, retomar las especificaciones de proceso o mini-especificaciones desarrolladas en el análisis, ya que el diseño de módulos, no es más que un refinamiento de la especificación de proceso elaborada anteriormente.[1]

Atributos de un modulo:

Un módulo consta de diferentes atributos, que se deben tener en cuenta para su adecuada definición:[1]

- *Entradas:* Son datos o parámetros de control que recibe el módulo de quien lo llama.

- *Salidas:* Son datos que entrega el módulo a quien lo llamó, una vez finalice su operación o mecánica.

- *Función:* Es la transformación que realiza el módulo de los datos de entrada en datos de salida.

- *Mecánica:* Es la lógica con que cada módulo ejecuta su función. Tiene cuatro elementos básicos: Condición, secuencia, repetición y rutina.

- *Datos Internos:* Es el conjunto de datos que utiliza internamente el módulo para realizar su mecánica.

Diseño de bases de datos.

Se debe establecer en detalle cómo será la estructura física, tablas, atributos, relaciones y formas de acceso de la información que almacenará el sistema. Es la última oportunidad que se tiene de refinar, corregir y definir la base de datos generada en el modelo de información, pues de esta actividad se desprende la construcción física de la base de datos.[1]

Una labor bien realizada aquí, garantiza con un alto grado de confiabilidad, que el sistema responderá al alcance planteado inicialmente, en forma precisa. Una mala definición de esta tarea, traerá como consecuencia el fracaso en la construcción del sistema, y el consecuente fracaso de la culminación del mismo.

### 2.2.2.2 Características en el diseño

Para determinar las medidas y posteriormente la métricas a aplicar en este flujo de trabajo es necesario conocer algunas características del diseño. Según Guerreo las principales características de un modulo en el diseño son el acoplamiento, la cohesión, el fan in, el fan out de un modulo.

- Acoplamiento.

El acoplamiento se define como el grado de interdependencia entre módulos. Es la medida de interacción de los módulos. Mide el grado de relaciones que existen entre los diferentes módulos de la estructura.[1]

Los buenos diseñadores buscan desarrollar la estructura de un sistema, de tal forma que un módulo tenga poca dependencia de cualquier otro módulo.

Los investigadores del diseño orientado a objeto expresan que un buen diseño debe proporcionar un acople mínimo, controlando variables como:[1]

- Número de parámetros que se transfieren entre módulos.
- Transferencia de datos innecesaria a los módulos que se llaman.
- Transferencias de datos, no información de control.

En el Anexo No 2 se puede apreciar el espectro de acople, donde se identifican bien las variables de acoplamiento.

La conectividad sencilla da como resultado un software fácil de mantener y menos propenso al “efecto onda” producido cuando los errores aparecen en una posición y se propagan a lo largo del sistema.[1]

Los niveles altos de acople, se producen cuando los módulos están ligados a un entorno externo. Por ejemplo, entradas de dispositivos, protocolos de comunicación. Aunque esto es esencial, debe limitarse a un pequeño número de módulos dentro de la estructura. [1]

- Cohesión.

La cohesión entre módulos es la medida con que cada módulo ejecuta su trabajo, evalúa al interior del módulo mientras que el acoplamiento evalúa interfaces entre

módulos. Se dice que a mayor cohesión, mejor es el diseño y se tiende a disminuir la comunicación o acople entre módulos.[1]

Un módulo cohesivo, ejecuta una tarea sencilla de un procedimiento de software y requiere poca interacción con procedimientos que ejecutan otra parte de un programa. En el Anexo No 2 se muestra como se puede manejar el espectro de cohesión.

Existen diferentes tipos de cohesión que se encuentran ordenados de mejor a peor: [1]

1. Cohesión Funcional: Se presenta cuando un módulo ejecuta una sola labor.

2. Cohesión Secuencial: Ocurre cuando un módulo ejecuta una labor de tal modo que la salida de una actividad, se convierta en la entrada para la siguiente actividad.

3. Cohesión de Comunicación: Ocurre cuando los componentes de un módulo toman la misma entrada y producen diferentes salidas. El orden de las componentes del módulo no es importante.

4. Cohesión Procedimental: Sucede cuando fluye control en el módulo. La secuencia de ejecución de las funciones es importante.

- Fan in.

El fan-in se define como el número de módulos que llaman a otro módulo, como se representa en la Figura 2.1. Un buen fan-in implica optimización en construcción, dado que se hace uso del mismo módulo, desde diferentes funciones que lo requieren y no se repite la construcción de la función en cada módulo.[1]

Un buen número de fan-in por módulo, puede ser tres, es decir lo anterior, que un módulo puede ser usado por máximo otros tres módulos diferentes.

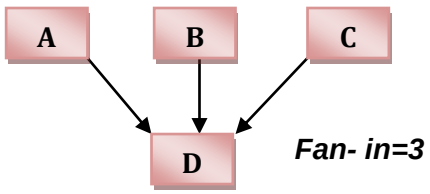



**Figura 2.1- Representación grafica del fan- in [1]**

- Fan out

En la Figura 2.2 se representa como el fan out es el número de módulos que son llamados por otro módulo. Un buen fan out implica un adecuado particionamiento de módulos a nivel de funciones específicas, lo cual mejora la cohesión en los módulos, ya que refuerza el hecho de evitar cohesiones como la procedimental y la de comunicación.

Se dice que un buen fan out es de nueve por módulo. Quiere decir, que un módulo puede llamar a otros nueve como buena medida de Fan Out. [1]

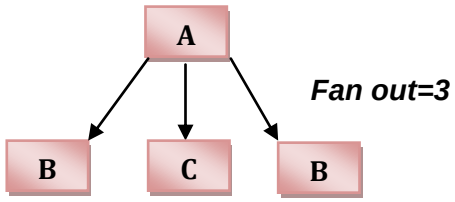



**Figura 2.2- Representación gráfica del fan out [1]**

Estas características antes vistas nos ayudan dentro del proceso de medición a determinar las medidas del flujo de trabajo análisis y diseño y posteriormente las métricas e indicadores que permiten evaluar estadísticamente este fuljo.

## *2.4 Medidas del Análisis y Diseño*

Después de haber realizado un estudio exhaustivo sobre el flujo de trabajo análisis y diseño propuesto por RUP en cuanto a sus características, actividades, y otros factores, se seleccionaron algunas de las medidas o atributos que se ponen de manifiesto en este flujo y que son la base para la aplicación de las métricas en el mismo.

Aunque se plantean las medidas de forma separada para un mejor entendimiento de la propuesta, recordar que en la investigación trabajamos el Análisis y Diseño de forma unida, como bien lo plantea la metodología RUP.

Medidas del Análisis

- Número de requisitos funcionales.
- Número de requisitos no funcionales.
- Número de requisitos totales.
- Número de requisitos de función únicos.

Medidas del Diseño

- Número de funciones que llaman a otra función.
- Número de funciones que son llamadas por una función.
- Expansión del módulo i.
- Número de variables de entrada y salida que entran y salen del módulo.
- Número total de módulos definidos en la arquitectura del programa.
- Número de elementos de base de datos.
- Número de segmentos de base de datos (registros diferentes u objetos individuales).
- Número de módulos con una salida y una entrada (el proceso de excepciones no se considera de salida múltiple).
- Número de referencias a variables y constantes que se utilizan en el módulo.
- Número de variables y constantes que afectan el valor de una variable dentro del módulo.
- Número de parámetros de datos de entrada y salida.
- Número de parámetros de control de entrada y salida.
- Número de variables globales usadas como datos.
- Número de variables globales usadas como control.
- Número de módulos llamados (expansión).
- Número de módulos que llaman al módulo en cuestión (concentración).
- Número de flujos de un módulo.
- Número de clases de otros paquetes que dependen de las clases del propio paquete.
- Número de clases dentro del propio paquete que dependen de clases de otros paquetes.
- Número de clases abstractas en un paquete.
- Número total de clases en un paquete.
- Número de nodos (módulos).
- Número máximo de nodos de cualquier nivel de la arquitectura.

Análisis y Diseño

- Número de defectos eliminados
- Número de escapes netos
- Número de errores encontrados

## *2.5 Propuesta de Métricas para Análisis y Diseño*

La propuesta como aporte científico al desarrollo de la calidad de software tiene la adaptación e interpretación de un conjunto de métricas antes definidas al flujo de trabajo Análisis y Diseño. La misma tiene como ventaja que los líderes de los proyectos que utilizan la metodología RUP puedan retroalimentar el proceso de evaluación de la calidad de los productos de software.

Para la realización de la propuesta de métricas primeramente se analizaron detenidamente todas las medidas seleccionadas en el epígrafe anterior y a partir de estas se consultaron bibliografías como por ejemplo la tesis "Las Métricas de Software y su Uso en la Región" de Heidia Dora Gonzáles desarrollada en México, principalmente el capítulo 4 donde aborda sobre las métricas en el desarrollo de un software y el libro de Pressman, para determinar las métricas e indicadores que nos brinden una visión estadística de la calidad del flujo de trabajo.

Para una correcta interpretación de la métrica se hace necesario tener conocimiento sobre las características y actividades que realiza el flujo de trabajo, para esto puedes apoyarte en el epígrafe anterior donde se conceptualizan muchas de estas características.

### 2.5.1 Métricas de la Calidad de Especificación

Existe una lista de características para poder valorar la calidad del modelo de análisis y la correspondiente especificación de requisitos: especificidad, corrección, compleción (estado completo/plenitud), comprensión, capacidad de verificación, consistencia externa e interna, capacidad de logro, concisión, trazabilidad, capacidad de modificación, exactitud y capacidad de reutilización. Muchas de las características anteriores pueden ser de naturaleza cuantitativa, y se pueden representar usando una o más métricas.

#### 2.5.1.1 Especificidad de los requisitos

Para evaluar la especificidad de los requisitos existe una métrica basada en la consistencia de la interpretación de los revisores para cada requisito, tal como:

$$Q_1 = \frac{nu_i}{n_i}$$

$$n_i = n_f + n_{nf} \qquad \textit{(2.1)}$$

Donde $Q_1$: Consistencia de la interpretación de los revisores para cada requisito.

$n_i$: Número de requisitos en una especificación.

$n_f$: Número de requisitos funcionales.

$n_{nf}$: Número de requisitos no funcionales.

$nu_i$: es el número de requisitos para los que todos los revisores tuvieron interpretaciones idénticas.

$0 \leq Q_1 \leq 1$ Cuanto más cerca de uno este el valor de $Q_1$ menor será la ambigüedad de la especificación.

### 2.5.1.2 Compleción de los requisitos funcionales

La compleción de los requisitos funcionales puede determinarse calculando la relación

$$Q_2 = \frac{n_u}{n_i * n_s} \qquad \textit{(2.2)}$$

Donde $Q_2$ : Número de funciones necesarias que se han especificado para un sistema

$n_u$: Número de requisitos de función únicos,

$n_i$: Número de entradas (estímulos) definidos o implicados por la especificación

$n_s$: Número de estados especificados.

$0 \leq Q_2 \leq 1$

La relación $Q_2$ contra más próximo este de 1 mayor va a ser el número de funciones que se han especificado para un sistema, sin embargo, no trata los requisitos no funcionales.

### 2.5.1.3 Grado de validación de los requisitos

Para incorporar los requisitos no funcionales se hace necesaria una métrica global completa, por lo que debemos considerar el grado de validación de los requisitos $Q_3$.

$$Q_3 = \frac{n_c}{n_c + n_{nv}} \qquad \textit{(2.3)}$$

Donde $Q_3$: Grado de validación de los requisitos.

$n_c$: Es el número de requisitos que se han validados como correctos.

$n_{nv}$: El número de requisitos que no se han validado todavía.

$0 \leq Q_3 \leq 1$, mientras más próximo este $Q_3$ de 1 mayor va a ser el grado de validación de los requisitos.

### 2.5.2 Métricas de diseño de alto nivel

Las métricas de diseño a alto nivel se concentran en las características de la estructura del programa dándole énfasis a la estructura arquitectónica y en la eficiencia de los módulos. Estas métricas son de caja negra, en el sentido de que no se requiere ningún conocimiento del trabajo interno de ningún modo en particular del sistema.

#### 2.5.2.1 Métricas de complejidad del software

Card y Glass proponen tres métricas de complejidad del software: complejidad estructural, complejidad de datos y complejidad del sistema. Las métricas de complejidad estructural y de datos se pueden aplicar al flujo de trabajo Análisis y Diseño y se puede realizar una estimación de la complejidad arquitectónica o del sistema.

La *complejidad estructural* $S(i)$, de un módulo i se define de la siguiente manera:

$$S(i) = f^2_{out}(i) \qquad \textit{(2.4)}$$

Donde $fout(i)$ es la expansión del módulo i. La expansión indica el número de módulos que son invocados directamente por el módulo i.

La *complejidad de datos* $D(i)$, proporciona una indicación de la complejidad en la interfaz interna de un módulo i y se define como:

$$D(i) = \frac{v(i)}{[fout(i) + 1]} \qquad \textit{(2.5)}$$

Donde $v(i)$ es el número de variables de entrada y salida del módulo i.

Finalmente se puede realizar una estimación de la *complejidad del sistema* $C(i)$, se define como la suma de las complejidades estructural y de datos, y se define como.

$$C(i) = S(i) + D(i) \qquad \textbf{(2.6)}$$

A medida que crecen los valores de complejidad, la complejidad arquitectónica o global del sistema también aumenta. Esto lleva a una mayor probabilidad de que aumente el esfuerzo necesario para la integración y las pruebas.

#### 2.5.2.2 Complejidad del flujo de información

$$IFC = \frac{fanin + fanout}{Tf} \qquad \textbf{(2.7)}$$

Donde $IFC$: Es la complejidad del flujo de información

$fanin$: Es la medida del número de funciones que llaman a otra función X

$fanout$: Es el número de funciones que son llamadas por una función X.

$Tf$: Totalidad de funciones, comprende también las que no llaman a otra función o no son llamadas por una función.

Para esta métrica debemos tener en cuenta el acoplamiento entre funciones dentro de un módulo (el acoplamiento mide el grado de relaciones que existen entre los diferentes módulos de la estructura), este fenómeno se explica en el subepígrafe 2.2.2.2. Debido a que un buen diseño debe proporcionar un acople mínimo, un valor aceptado del $IFC$ son los valores que se encuentren entre 0 y 0,5. El valor óptimo que puede tomar el software es 0,5.

#### 2.5.2.3 Métricas de morfología simples

Existen varias *métricas de morfología* simples (por ejemplo, forma) que permiten comparar diferentes arquitecturas de programa mediante un conjunto de dimensiones directas.

En la Figura 2.3, se muestra un ejemplo de una arquitectura de software, donde puede visualizarse la estructura de la arquitectura.

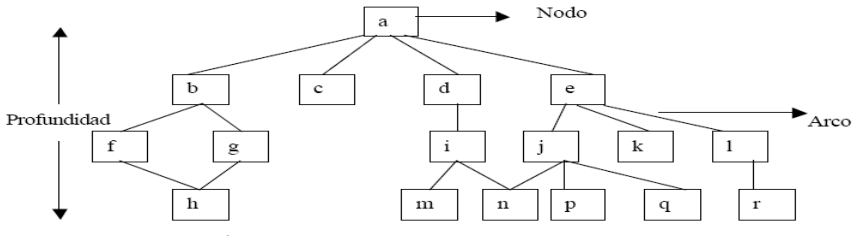



**Figura 2.3- Arquitectura de software**

Para determinar el tamaño, la profundidad y la anchura de la arquitectura de software tenemos la siguiente propuesta de métrica basándonos en las métricas planteadas en la tesis de Heidia Doria González [4].

$$T = N + A \qquad \textit{(2.8)}$$

Donde T: Es el tamaño de la arquitectura

N: Es el número de nodos (módulos)

A: Es el número de arcos (líneas de control)

**Profundidad** (P) = el camino más largo desde el nodo raíz (parte más alta) a un nodo hoja.

**Anchura** (A)= máximo número de nodos de cualquier nivel de la arquitectura.

Podemos medir la densidad de conectividad de la arquitectura mediante la siguiente métrica:

$$R = \frac{A}{N} \qquad \textit{(2.9)}$$

Donde: R es la relación arco a nodo

A es el número de arcos (líneas de control)

N es el número de nodos (módulos)

La relación arco a nodo (R) proporciona una indicación del acoplamiento de la arquitectura.

### 2.5.2.4 Índice de calidad de la estructura del diseño (ICED)

La Fuerza Aérea Americana ha desarrollado varios indicadores de calidad del software basados en características medibles del diseño de un programa de computadora. Empleando conceptos similares, la Fuerza Aérea utiliza información obtenida del diseño de datos y arquitectónico para obtener un *índice de calidad de la estructura del diseño* (ICED)[4]

Se deben comprobar los siguientes valores para calcular el ICED[4]:

$S_1$ = el número total de módulos definidos en la arquitectura del programa

$S_2$ = el número de módulos que para funcionar correctamente dependen de la fuente de datos de entrada o producen datos que se van a emplear en algún otro lado en general, los módulos de control (entre otros) no formarían parte de S2.

$S_3$ = el número de módulos que para que funcionen correctamente dependen de procesos previos

$S_4$ = el número de elementos de base de datos (incluye todos los objetos de datos y todos los atributos que definen los objetos)

$S_5$ = el número total de elementos únicos de base de datos

$S_6$ = el número de segmentos de base de datos (registros diferentes u objetos individuales)

$S_7$ = el número de módulos con una salida y una entrada (el proceso de excepciones no se considera de salida múltiple)

Una vez que se han determinado los valores $S_1$ a $S_7$ para un software, se pueden calcular los siguientes valores intermedios:

*Estructura del programa*

$D_1$ se define como sigue: si el diseño arquitectónico se desarrolló usando un método determinado (por ejemplo, diseño orientado a flujo de datos o diseño orientado a objetos), entonces $D_1$=1, de otra manera $D_1$=0.

*Independencia del módulo*

$$D_2 = 1 - \left(\frac{S_2}{S_1}\right) \qquad \textbf{(2.10)}$$

*Módulos no dependientes de procesos previos*

$$D_3 = 1 - \left(\frac{S_3}{S_1}\right) \qquad \textbf{(2.11)}$$

*Tamaño de la base de datos*

$$D_4 = 1 - \left(\frac{S_5}{S_4}\right) \qquad \textbf{(2.12)}$$

*Compartido de la base de datos*

$$D_5 = 1 - \left(\frac{S_6}{S_4}\right) \qquad \textbf{(2.13)}$$

*Características de entrada/salida del módulo*

$$D_6 = 1 - \left(\frac{S_7}{S_1}\right) \qquad \textit{(2.14)}$$

Con estos valores intermedios calculados, el ICED se obtiene de la siguiente manera:

$$ICED = \sum_{i=1a6} Wi\ Di \qquad \textit{(2.15)}$$

Donde: i = 1 a 6

$W_i$ es el peso relativo de la importancia de cada uno de los valores intermedios y $W_i$ =1 (si todos los $D_i$ tiene el mismo peso, entonces $W_i$ =0.167).

$$0 \leq ICED \leq 1$$

Se puede determinar y compartir el valor de *ICED* de diseños anteriores con un diseño actualmente en desarrollo. Si el *ICED* es significativamente inferior a la media, se aconseja seguir trabajando en el diseño y revisión. Igualmente, si hay que hacer grandes cambios a un diseño ya existente, se puede calcular el efecto de estos cambios en el *ICED***.**

### 2.5.3 Métricas de diseño en los componentes

Las métricas de diseño a nivel de componentes se concentran en las características internas de los componentes del software e incluyen medidas de la cohesión, acoplamiento y complejidad del módulo. Estas tres medidas pueden ayudar al desarrollador de software a juzgar la calidad de un diseño a nivel de componentes.

Las métricas presentadas son de “caja blanca” en el sentido de que requieren conocimiento del trabajo interno del módulo en cuestión. Las métricas de diseño en los componentes se pueden aplicar una vez que se ha desarrollado un diseño procedimental. También se pueden retrasar hasta tener disponible el código fuente.

#### 2.5.3.1 Métricas de cohesión.

Para medir la cohesión de un módulo es necesario definir varios conceptos.

- número de elementos (tokens)

Son todas las referencias a variables y constantes que se utilizan en el módulo. Si una variable aparece varias veces, cada una se toma como un elemento. Para distinguirlos se les puede agregar como subíndice un número que dice si es la primera aparición, la segunda u otra.

- rebanada de datos (data slice)

Es el conjunto de variables y constantes que afectan el valor de una variable dentro del módulo. Se forma como sigue:

a) se comienza por una variable de resultado (ya sea regreso de una función o variable de salida), usualmente al final, donde guarda su valor definitivo y se agrega a la rebanada, incluyendo el subíndice que le tocó al revisar los elementos.

b) Se analiza el lado derecho de la asignación y se anotan las variables y constantes que intervienen en el cálculo de su valor.

c) Para cada elemento de la rebanada, se revisan las líneas anteriores disminuyendo una a una, repitiendo el proceso del paso b.

- adhesivo (glue)

Se le llamará adhesivo a un elemento que aparece en dos o más rebanadas.

- superadhesivo (superglue).

Se denomina superadhesivo a un elemento que está en todas las rebanadas de un módulo.

Con todas estas definiciones podemos calcular la cohesión funcional fuerte y débil de un módulo, mediante las siguientes métricas:

Cohesión funcional fuerte

$$CFF = \frac{N_s}{N_e} \qquad \textbf{(2.16)}$$

Donde *CFF:* Es la cohesión funcional fuerte

$N_s$: Es el número de superadhesivo

$N_e$: Es el número de elementos

$0 \leq CFF \leq 1$

*Cohesión Funcional Débil*

$$CFD = \frac{N_n}{N_e} \qquad \textbf{(2.17)}$$

Donde *CFD:* Es la cohesión funcional débil

$N_a$: Es el número de adhesivos

$N_e$: Es el número de elementos

$0 \leq CFD \leq 1$

Tanto la métrica de cohesión funcional fuerte como débil poseen valores que van desde 0 a 1, mientras más cercano sean *CFF* ó *CFD* a 1 mayor será la cohesión del módulo, tienen un valor de 0 cuando un procedimiento tiene más de una salida y no muestra ningún atributo de cohesión indicado por una métrica particular.

Un procedimiento sin señales de súper-unión, sin señales comunes a todas las porciones de datos, no tiene una cohesión funcional fuerte (no hay señales de datos que contribuyan a todas las salidas) Un procedimiento sin señales de unión, es decir, sin señales comunes a más de una porción de datos (en procedimientos con más de una porción de datos), no muestra una cohesión funcional débil y ninguna adhesividad (no hay señales de datos que contribuyan a más de una salida). La cohesión funcional fuerte y la pegajosidad se obtienen cuando las métricas toman un valor máximo de 1.

### 2.5.3.2 Métricas de acoplamiento

El acoplamiento de módulo proporciona una indicación de la "conectividad" de un módulo con otros módulos, datos globales y entorno exterior. Proponemos una métrica para el acoplamiento del módulo que combina el acoplamiento de flujo de datos y de control: acoplamiento global y acoplamiento de entorno. Las medidas necesarias para calcular el acoplamiento de módulo se definen en términos de cada uno de los tres tipos de acoplamiento apuntados anteriormente. [4]

*Para el acoplamiento de flujo de datos y de control*

$d_i$ = número de parámetros de datos de entrada

$c_i$ = número de parámetros de control de entrada

$d_o$ = número de parámetros de datos de salida

$c_o$ = número de parámetros de control de salida

*Para el acoplamiento global*

$g_d$ = número de variables globales usadas como datos

$g_c$ = numero de variables globales usadas como control

*Para el acoplamiento de entorno:*

$w$ = número de módulos llamados (expansión)

$r$ = número de módulos que llaman al módulo en cuestión (concentración)

Usando estas medidas se define un indicador de acoplamiento de módulo ($m_c$), de la siguiente manera:

$$m_c = \frac{K}{M} \qquad \textit{(2.18)}$$

Donde $m_c$: Es el indicador de acoplamiento de módulo

$K$=1 constante de proporcionalidad

$$M = d_i + a * c_i + d_o + b * c_o + g_d + c * g_c + w + r$$

Donde: $a = b = c = 2$

El valor de $m_c$ oscila entre 0 y 1 ($0 \leq mc \leq 1$) y, a mayor valor de $m_c$ menor es el acoplamiento de módulo.

Para que aumente la métrica de acoplamiento a medida que aumenta el grado de acoplamiento, se puede definir una métrica de acoplamiento revisada (*C)*

$$C = 1 - m_c \qquad \textit{(2.19)}$$

Donde el grado de acoplamiento no aumenta linealmente, un valor mínimo en el rango de 0,66 hasta un máximo que se aproxima a 1.

### 2.5.4 Métricas para medir la interdependencia entre subsistemas del diseño

R. Martin en 1994 describió un conjunto de métricas para medir la calidad de un diseño software en términos de interdependencia entre subsistemas. Los diseños con alta interdependencia entre subsistemas son: Rígidos, No reutilizables, Difícil de mantener

Proponemos un conjunto de métricas para buscar estas propiedades del sistema la responsabilidad, independencia y estabilidad de un subsistema se puede cuantificar contando el número de dependencias que interactúan con el subsistema

#### 2.5.4.1 Métrica para la estabilidad de un subsistema

Se propone la siguiente métrica para la estabilidad de un subsistema:

$$I = \frac{C_e}{C_a + C_e} \qquad \textit{(2.20)}$$

Donde: *I* es la estabilidad del subsistema

*Ca* (*Afferent Couplings*): Número de clases de otros paquetes que dependen de las clases del propio paquete. Indicador de responsabilidad del propio paquete

*Ce* (*Efferent Couplings*): Número de clases dentro del propio paquete que dependen de clases de otros paquetes.

($0 \leq I \leq 1$), Cuando toma el valor de 1 hay una máxima inestabilidad en el subsistema y cuando toma el valor de 0 hay una máxima estabilidad en el subsistema por lo que los sistemas son incambiables e inflexibles y no pueden hacer uso de otros subsistemas. El diseño debe ser flexible para soportar futuros cambios

¿Cómo un subsistema con estabilidad máxima *(I=0)* puede ser flexible para soportar cambios?

- Las clases abstractas proporcionan suficiente flexibilidad para ser extendidas con nuevos requerimientos en subsistemas externos

Por lo que llegamos a la conclusión que los subsistemas estables deberían ser altamente abstractos mientras que los inestables deberían ser altamente concretos.

### 2.5.4.2 Métrica para la abstracción de un paquete

La abstracción es un mecanismo que permite al diseñador centrarse en los detalles esenciales de algún componente de un programa (tanto si es un dato como si es un proceso) sin preocuparse por los detalles de nivel inferior. Cuando los niveles de abstracción van elevándose, se ignoran más y más detalles, por lo tanto, se proporciona una visión más general de un concepto u objeto. A medida que pasamos a niveles más reducidos de abstracción, se muestran más detalles, esto es, se proporciona una visión más específica de un concepto u objeto.

$$A = \frac{NAC}{NTC} \qquad \textbf{(2.21)}$$

Donde *A:* Es la abstracción de un paquete

*NAC:* Número de clases abstractas en un paquete

*NTC:* Número total de clases en un paquete

($0 \leq A \leq 1$), Cuando toma el valor de 1 el paquete es completamente abstracto y cuando toma el valor de 0 el paquete es completamente concreto.

*Relación entre estabilidad (I) y abstracción (A)*

- Casos Ideales

(0,1) Subsistema con estabilidad máxima y nivel de abstracción máximo

(1,0) Subsistema con estabilidad mínima y nivel de abstracción mínimo

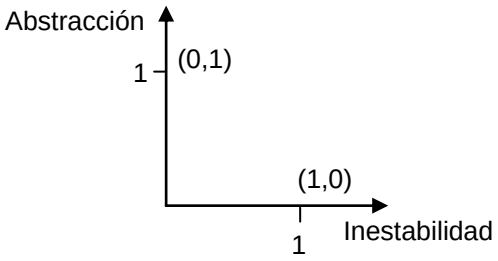



**Figura 2.4- Relación entre estabilidad y abstracción**

Otros casos que muestran la relación entre estabilidad y abstracción se demuestran en el Anexo No 3

### 2.5.5 Métrica para la probabilidad de cambios en el modelo

Probabilidad de cambios en el modelo

$$X = \frac{A}{B} \qquad \textbf{(2.22)}$$

Donde *X*= Registrabilidad de cambios en el diseño

*A*= número de cambios a funciones que tienen comentarios confirmados

*B* = total de funciones modificados

($0 \leq X \leq 1$), Entre más cercano a 1 más registrable, 0 indica un control de cambios deficiente o pocos cambios y alta estabilidad en el modelo.

### 2.5.6 Métrica de eficacia de la eliminación de defectos

Una métrica de la calidad que proporciona grandes beneficios a un proyecto, es la eficacia de la eliminación de defectos (EED). Cuando manipula dentro del proyecto, para evaluar la habilidad de un equipo en encontrar errores antes de que pasen a la siguiente actividad, estructura o tarea de ingeniería del software. Por ejemplo, la tarea de análisis de requerimientos produce un modelo de análisis qué se puede inspeccionar para encontrar y corregir errores. Esos errores que no se encuentren durante la revisión del modelo de análisis se pasan a la tareas de diseño (en donde se puede encontrar o no). [4]

La métrica de eficacia de la eliminación de defecto se puede calcular tanto en el análisis como en el diseño de un software.

La EED lo podemos calcular mediante la siguiente métrica:

$$EED = \frac{E_i}{E_i + E_{i+1}} \qquad \textbf{(2.23)}$$

Donde *EED:* Es la eficacia de la eliminación de defectos

$E_i$*:* Es el número de errores encontrados durante la actividad *i-esima* de: ingeniería del software *i*

$E_{i+1}$*:* Es el número de errores encontrado durante la actividad de ingeniería del software (i + 1) que se puede seguir para llegar a errores que no se detectaron en la actividad *i.*

Un objetivo de calidad de un equipo de ingeniería de software es alcanzar un *EED* que se aproxime a 1, esto es, los errores se deberían filtrar antes de pasarse a la siguiente actividad.

### 2.5.7 Métrica de rendimiento

Otra 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. Con el resumen de defectos, no se observa el avance del rendimiento, solamente su valor final. Sin embargo el método de calcularlo es mucho más fácil, el rendimiento por cada etapa se podrá calcular por:

$$R = \frac{D_e * 100}{D_e + E_n} \qquad \textbf{(2.24)}$$

Donde *R:* Es el rendimiento en la etapa

$D_e$: Son los defectos eliminados

$E_n$: son los escapes netos. Los escapes netos son los defectos que se insertaron antes o durante una etapa determinada, pero que no fueron encontrados en esa etapa, sino en otra etapa posterior.

Debe quedar claro que un valor de rendimiento alto es mucho mejor que un valor bajo, el cual es un resultado pobre. El objetivo o meta es un rendimiento de 100%, o sea que se traten de eliminar los defectos tan pronto como sea posible.

## 2.6 Resumen de Métricas

| | Indicador | Fórmula | Interpretación | Referencia |
|---|---|---|---|---|
| **Métricas de la Calidad de Especificación** | $Q_1$: Consistencia de la interpretación de los revisores para cada requisito | $Q_1 = \frac{nu_i}{n_i}$<br>$n_i = n_f + n_{nf}$ | Métrica que indica la ambigüedad en la interpretación de los revisores.<br>$0 \leq Q_1 \leq 1$<br>Cuanto más cerca de uno este el valor de $Q_1$ menor será la ambigüedad de la especificación. | Ref-2.1 |
| | $Q_2$ : Número de funciones necesarias que se han especificado para un sistema | $Q_2 = \frac{n_u}{n_i * n_s}$ | Métrica que indica cuán completa (compleción) está la especificación de los requisitos funcionales.<br>$0 \leq Q_2 \leq 1$<br>Mientras más próximo este de 1 mayor va a ser el número de funciones que se han especificado para un sistema | Ref-2.2 |
| | $Q_3$: Grado de validación de los requisitos | $Q_3 = \frac{n_c}{n_{c+n_{nv}}}$ | Métrica que incorpora los requisitos no funcionales y considera el grado de validación de los requisitos.<br>$0 \leq Q_3 \leq 1$<br>Mientras más próximo este $Q_3$ de 1 mayor va a ser el grado de validación de los requisitos. | Ref-2.3 |
| **Métricas de diseño de alto nivel** | $S(i)$: Complejidad estructural de un módulo i<br>$D(i)$: Complejidad de datos de un módulo i<br>$C(i)$: Complejidad del sistema | $S(i) = f^2{}_{out}(i)$<br>$D(i) = \frac{v(i)}{[fout(i)+1]}$<br>$C(i) = S(i) + D(i)$ | Las métricas indican la complejidad del software.<br>A medida que crecen los valores de complejidad estructural y de datos, aumenta el valor de la complejidad del sistema. | Ref-2.4, 2.5 y 2.6 |

| | | | | |
|---|---|---|---|---|
| | $IFC$: Complejidad del flujo de información | $IFC = \frac{fanin + fanout}{Tf}$ | Métrica que indica la complejidad del flujo de información. Hay que tener en cuenta el acoplamiento entre funciones dentro de un modulo.<br>Un buen diseño debe proporcionar un acople mínimo.<br>Un valor aceptado del $IFC$ son los valores que se encuentren entre 0 y 0,5. El valor óptimo que puede tomar el software es 0,5. | Ref-2.7 |
| | T: Tamaño de la arquitectura<br>R: Relación arco a nodo | $T = N + A$<br>$R = \frac{A}{N}$ | Métricas de morfología simples que indican el tamaño y la densidad de conectividad de la arquitectura.<br>La relación arco a nodo (R) proporciona una indicación del acoplamiento de la arquitectura. | Ref-2.8 y 2.9 |
| | ICED: Índice de calidad de la estructura del diseño | $ICED = \sum_{i=1a6} Wi\ Di$ | $0 \leq ICED \leq 1$<br>Si el ICED es significativamente inferior a la media (0.5), se aconseja seguir trabajando en el diseño | Ref-2.10, 2.11, 2.12, 2.13, 2.14, 2.15 |
| **Métricas de diseño en los componentes** | CFF: Cohesión funcional fuerte<br>CFD: Cohesión funcional débil | $CFF = \frac{N_s}{N_e}$<br>$CFD = \frac{N_n}{N_e}$ | Métricas que indican la cohesión de un módulo.<br>$0 \leq CFF \leq 1$ $0 \leq CFD \leq 1$<br>Mientras más cercano sean *CFF* ó *CFD* a 1 mayor será la cohesión del módulo, tienen un valor de 0 cuando un procedimiento tiene más de una salida y no muestra ningún atributo de cohesión indicado por una métrica particular. | Ref-2.16 y 2.17 |
| | $m_c$: Acoplamiento de módulo<br>C: Grado de acoplamiento | $m_c = \frac{K}{M}$<br>$C = 1 - m_c$ | Métrica de acoplamiento que proporciona indicación de la "conectividad" de un módulo con otros módulos, datos globales y entorno exterior<br>$0 \leq m_c \leq 1$<br>Mientras más próximo a 1 sea $m_c$ menor es el acoplamiento de módulo.<br>C debe tomar un valor mínimo en el rango de 0,66 hasta un máximo que se aproxima a 1. | Ref-2.18 y 2.19 |

| | | | | |
|---|---|---|---|---|
| **Métricas para medir la interdependencia entre subsistemas del diseño** | I: Estabilidad del subsistema | $I = \frac{C_e}{C_a + C_e}$ | Métrica que indica la estabilidad de un subsistema<br>$0 \leq I \leq 1$<br>Cuando toma el valor de 1 hay una máxima inestabilidad en el subsistema y cuando toma el valor de 0 hay una máxima estabilidad en el subsistema por lo que los sistemas inflexibles y no pueden hacer uso de otros subsistemas. | Ref-2.20 |
| | A: Abstracción de un paquete | $A = \frac{NAC}{NTC}$ | La abstracción permite al diseñador centrarse en los detalles esenciales de algún componente de un programa.<br>$0 \leq A \leq 1$<br>Cuando toma el valor de 1 el paquete es completamente abstracto y por tanto proporciona una visión más general de un concepto u objeto y cuando toma el valor de 0 el paquete es completamente concreto y proporciona una visión más específica. | Ref-2.21 |
| **Métrica para la probabilidad de cambios en el diseño** | X= Registrabilidad de cambios en el diseño | $X = \frac{A}{B}$ | $0 \leq X \leq 1$<br>Entre más cercano a 1 más registrable, 0 indica un control de cambios deficiente o pocos cambios y alta estabilidad en el modelo. | Ref-2.22 |
| **Métrica de eficacia de la eliminación de defectos** | EED: Eficacia de la eliminación de defectos | $EED = \frac{E_i}{E_i + E_{i+1}}$ | 0 ≤ EED ≤ 1<br>Mientras más próximo a 1 se encuentre la EED, los errores encontrados se deberían filtrar antes de pasar a la siguiente actividad. | Ref-2.23 |

| | | | | |
|---|---|---|---|---|
| **Métrica de rendimiento** | R*:* Rendimiento en la etapa | $R = \frac{D_e * 100}{D_e + E_n}$ | Debe quedar claro que un valor de rendimiento alto es mucho mejor que un valor bajo, el cual es un resultado pobre. El objetivo o meta es un rendimiento de 100%, o sea que se traten de eliminar los defectos tan pronto como sea posible. | Ref-2.24 |

**Tabla 2.1 -** Resumen de las métricas propuestas

## *2.7 Conclusiones del Capítulo*

En este capítulo presentamos las principales características así como las actividades que se realizan en el flujo de trabajo de análisis y diseño de un software. Apoyándonos en el proceso de medición descrito en subepígrafe 1.3.1 se realizó una selección de las medidas en base de estas características y posteriormente se elaboró la propuesta de métricas, se y realizó la interpretación de los indicadores derivados de las métricas propuestas.

# Capítulo 3

# Validación de la propuesta

## 3.1 Introducción

En el capítulo anterior realizamos la descripción de la propuesta, la cual en este capítulo será validada por un comité de experto. Para la evaluación técnica de la propuesta emplearemos un método cuantitativo que tiene como fundamento la evaluación por parte de expertos en el tema de criterios anteriormente definidos. En el transcurso de este capítulo describiremos el método a aplicar y se mostraran los resultados derivados de la evaluación.

## 3.2 Método para la validación de la propuesta

Para realizar la evaluación técnica de la propuesta utilizamos el método de experto, el cual te permite tomar decisiones para aceptar o no la propuesta de acuerdo con los criterios definidos. [12]

Para llevar a cabo el buen desarrollo del mismo nos apoyamos en la tesis de Karenia Donatien Goliath y Yudermis Rodríguez Martínez realizada en la Universidad de las Ciencias Informáticas en Julio del 2007, donde efectúan un conjunto de pasos:

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

   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) Uso de la metodología RUP

   Peso: _____

8) Los ingenieros de software deben de tener conocimiento sobre el flujo de trabajo análisis y diseño propuesto por la metodología RUP.

   Peso: _____

9) Los ingenieros de software no deben de tener conocimiento del proceso de medición ni de las métricas para ser aplicadas.

   Peso: _____

Grupo No.3: Criterios de flexibilidad

10) Adaptabilidad a proyectos productivos que utilicen la metodología RUP.

    Peso: _____

11) Uso de las métricas necesarias para retroalimentar el proceso de evaluación de la calidad en los proyectos productivos orientado a objeto.

    Peso: _____

12) Facilidad de entendimiento de las métricas.

    Peso: _____

13) Flexibilidad en la interpretación de los resultados de las métricas aplicadas.

    Peso: _____

Grupo No.4: Criterios de impacto

14) Repercusión en los proyectos productivos que utilicen la metodología RUP.

    Peso: _____

15) Aceptación de la propuesta por los líderes de proyecto.

    Peso: _____

16) Organización en el proceso de aplicación de la métrica.

    Peso: _____

17) Impacto en el área a la cual está destinada.

Peso: _____

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 valore el peso relativo de cada criterio y así poder calcular la concordancia entre los expertos, Anexo No 4, y otro para calcular el nivel de aceptación de la propuesta con una escala de 1-5 y la apreciación cualitativa con una clasificación final de la propuesta en excelente, bueno, aceptable, cuestionable y malo. También se da la posibilidad de dar su opinión haciendo una valoración final de la propuesta, emitiendo todas aquellas consideraciones que estimaron convenientes, Anexo No 5.

5. Para calcular la concordancia en el trabajo de los expertos después de recibir los valores del peso relativo de cada criterio, se construye la Tabla 3.1, donde:

E: es el número de expertos que realizan la evaluación
C: es el número de criterios que son evaluados.
G: es el número del grupo al que pertenecen los criterios

| G | C/E | $E_1$ | $E_2$ | $E_3$ | $E_4$ | $E_5$ | $E_6$ | $E_7$ | $E_p$ |
|---|---|---|---|---|---|---|---|---|---|
| 25 | $C_1$ | | | | | | | | |
| | $C_2$ | | | | | | | | |
| | $C_3$ | | | | | | | | |
| | $C_4$ | | | | | | | | |
| | $C_5$ | | | | | | | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **30** | **$C_6$** | | | | | | | | |
| | **$C_7$** | | | | | | | | |
| | **$C_8$** | | | | | | | | |
| | **$C_9$** | | | | | | | | |
| **20** | **$C_{10}$** | | | | | | | | |
| | **$C_{11}$** | | | | | | | | |
| | **$C_{12}$** | | | | | | | | |
| | **$C_{13}$** | | | | | | | | |
| **25** | **$C_{14}$** | | | | | | | | |
| | **$C_{15}$** | | | | | | | | |
| | **$C_{16}$** | | | | | | | | |
| | **$C_{17}$** | | | | | | | | |
| **T** | | | | | | | | | |

Tabla 3.1- Resumen de la evaluación emitida por los expertos.

6. Se utiliza el coeficiente de concordancia de Kendell y el estadígrafo Chi cuadrado (X2) para verificar la consistencia en el trabajo de los expertos, para esto se sigue con el siguiente procedimiento.

   - 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

     ∆C: 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

$$X^2 = E\,(C\text{-}1)\,W$$

Los valores obtenidos del cálculo de la concordancia de Kendall se muestran en la Tabla 3.2

| **Expertos/Criterios** | **$E_1$** | **$E_2$** | **$E_3$** | **$E_4$** | **$E_5$** | **$E_6$** | **$E_7$** | **$\sum E$** | **$E_p$** | **$\Delta C$** | **$\Delta C^2$** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **$C_1$** | | | | | | | | 0 | 0 | 0 | 0 |
| **$C_2$** | | | | | | | | 0 | 0 | 0 | 0 |
| **$C_3$** | | | | | | | | 0 | 0 | 0 | 0 |
| **$C_4$** | | | | | | | | 0 | 0 | 0 | 0 |
| **$C_5$** | | | | | | | | 0 | 0 | 0 | 0 |
| **$C_6$** | | | | | | | | 0 | 0 | 0 | 0 |
| **$C_7$** | | | | | | | | 0 | 0 | 0 | 0 |
| **$C_8$** | | | | | | | | 0 | 0 | 0 | 0 |
| **$C_9$** | | | | | | | | 0 | 0 | 0 | 0 |
| **$C_{10}$** | | | | | | | | 0 | 0 | 0 | 0 |
| **$C_{11}$** | | | | | | | | 0 | 0 | 0 | 0 |
| **$C_{12}$** | | | | | | | | 0 | 0 | 0 | 0 |
| **$C_{13}$** | | | | | | | | 0 | 0 | 0 | 0 |
| **$C_{14}$** | | | | | | | | 0 | 0 | 0 | 0 |
| **$C_{15}$** | | | | | | | | 0 | 0 | 0 | 0 |
| **$C_{16}$** | | | | | | | | 0 | 0 | 0 | 0 |
| **$C_{17}$** | | | | | | | | 0 | 0 | 0 | 0 |
| **DC** | 0 | | | | | | | | | | |
| **M$\sum$E** | 0 | | | | | | | | | | |
| **W** | 0 | | | | | | | | | | |
| **$X^2$** | 0 | | | | | | | | | | |

Tabla 3.2- Tabla resumen para el cálculo de concordancia de Kendall

- El Chi cuadrado calculado se compara con el obtenido del las tablas estadísticas

  Si se cumple:

$$X^2_{real} < X^2_{(\alpha,\ c\text{-}1)}$$

  Existe concordancia en el trabajo de expertos.

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

Una vez comprobada la consistencia del trabajo de expertos se puede determinar el nivel de aceptación de la propuesta entre los expertos, para esto debemos seguir los siguientes pasos.

1. Para determinar el nivel de aceptación debemos definir el peso relativo de cada criterio (P).
2. Conociendo el peso de cada criterio y la calificación dada por los evaluadores en una escala de 1-5 se puede construir la Tabla 3.3, para obtener el valor de de

   P × c, donde (c), es el criterio promedio concebido por los expertos.

| Criterios | Clasificación (c) | | | | | P | P x c |
|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | | |
| C1 | | | | | | | |
| C2 | | | | | | | |
| C3 | | | | | | | |
| C4 | | | | | | | |
| C5 | | | | | | | |
| C6 | | | | | | | |
| C7 | | | | | | | |
| C8 | | | | | | | |
| C9 | | | | | | | |
| C10 | | | | | | | |
| C11 | | | | | | | |
| C12 | | | | | | | |
| C13 | | | | | | | |
| C14 | | | | | | | |
| C15 | | | | | | | |
| C16 | | | | | | | |
| C17 | | | | | | | |

Tabla 3.3- Resumen de la clasificación de cada criterio

3. Se calcula el Índice de Aceptación del proyecto (IA).

   IA = Σ (P × c) / 5

4. Por último se determina la probabilidad de éxito de la propuesta y para esto debemos conocer los siguientes **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

## *3.3 Análisis de la evaluación técnica de propuesta*

Para la validación de la propuesta se seleccionaron 7 expertos, teniendo en cuenta su currículo, experiencia laboral y área a la que pertenece en estos momentos.

A cada experto se les entregó una encuesta con dos modelos, para que formularan su opinión dándole peso a cada criterio, con estos valores se construyó la tabla de peso relativo de cada criterio, Anexo No 6.

Luego se calculó la concordancia entre los expertos, Anexo No 7, con los valores de la tabla anterior, lo que dio como resultado:

$X^2$ real es 26.7568, para seleccionar el $X^2$ de la tabla de Distribución Chi Cuadrado, Anexo No 8, se toma 1-α=0.99 dónde α es el error permisible, entonces α=0.01. Debe cumplirse que $X^2 < X^2$ (α, c-1).

El cálculo arrojó como resultado:

26.7568 < 31.9999, por lo que se llega a la conclusión de que existe concordancia entre los expertos y se puede pasar a la construcción de la tabla de clasificación de cada criterio para saber el índice de aceptación que tuvo la propuesta, Anexo No 9.

Una vez que estén los datos en la tabla se calcula el Índice de Aceptación (IA) que sería: 0.73852, el cual se compara con los valores que aparecen a continuación.

**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

Se puede concluir que la propuesta tiene una alta probabilidad de éxito.

## *3.4 Conclusiones*

En el presente capítulo se realizó la evaluación de la propuesta de solución para ello se utilizó el método de experto. Al aplicar el método y analizar los resultados se obtuvo una alta probabilidad de éxito por lo que la aplicación de la propuesta debe brindar resultados favorables.

# Conclusiones Generales

A partir de todo el trabajo realizado, se obtuvo la propuesta, la cual recoge la interpretación de varias métricas que permiten evaluar el flujo de trabajo Análisis y Diseño.

Para ello se investigó sobre la calidad, su control y evaluación y si en la UCI se estaban aplicando métricas para el flujo de trabajo Análisis y Diseño. Además se estudió detalladamente el flujo de trabajo y se le aplicaron métricas ya existentes, elaborando una detallada y fácil interpretación de las mismas.

Para la validación de esta investigación se aplicó el criterio de expertos el cual arrojó que la probabilidad de éxito es alta, lo que implica desde el punto de vista teórico, el cumplimiento de la idea a defender planteada. Hay que aclarar que la propuesta solo es adaptable al software que utilice la metodología RUP.

# Recomendaciones

Se recomienda para próximas investigaciones:

- Profundizar en el estudio y enriquecimiento de la propuesta.
- Analizar métricas más sencillas por la complejidad al utilizarlas.
- Los ingenieros de software que aplicarán la propuesta deben de tener conocimiento del proceso de medición y de las métricas para ser aplicadas en los proyectos de la UCI que utilicen la metodología RUP.
- Extender la propuesta a otras metodologías de desarrollo de software.
- Proponer una plantilla de recolección de datos para ser usados en los cálculos de las métricas.
- Aplicar la propuesta en proyectos productivos reales.

# Referencias Bibliográficas


1. Peña, D.G., *Elementos básicos de ingeniería del software*. Septiembre de 2007.
2. Reynoso, C.B., *Introducción a la Arquitectura de Software*. Marzo, 2004.
3. Rubio, G.B., *Calidad en Ingeniería del Software*. 2002.
4. Gonzales, H.D., *Las Métricas de Software y su Uso en la Región*, in *Departamento de Ingeniería en Sistemas Computacionales*. Mayo, 2001, Universidad de las Américas Puebla.
5. Oscar M. Fernández, D.G., Alfa Beltrán *Un enfoque actual sobre la calidad del software*. 1995
6. Mojena, B.G., *Procedimiento Propuesto para medir la Calidad en la Gestión de Requisitos*. Julio 2007.
7. José Miguel Calvo Medrano, J.M.M.M., *Medida de las subcaracterísticas Capacidad de Análisis y Capacidad de Cambio mediante la norma ISO/IEC 9126.*
8. Escorial, J.S., *La Norma ISO/IEC 9126.*
9. Scalone, F., *Auditoría de la Calidad del Software* Octubre, 2006.
10. *Goal Question Metric*. [Cited; available from: www.fing.edu.uy/inco/cursos/gestsoft/Presentaciones/GQM%20-%20G9/GQM.ppt
11. GALLEGO, J.P.G., *FUNDAMENTOS DE LA METODOLOGIA RUP.* 2007.
12. Karenia Donatien Goliath, Y.R.M., *Documentación imprescindible para los flujos de trabajo de diseño e implementación de software de gestión*. 2007: Cuba.
13. Pressman, R.S., *Ingeniería del Software. Un enfoque práctico* Quinta ed.
14. Dayami Rodríguez Brito, H.F.D., *Análisis del Método de Estimación empleado para el desarrollo del proyecto SIGEP*. Julio 2007: Cuba.
15. Velthuis, M.P. *Métricas*. 2007
16. Escorial, J.S., *La medida en la Calidad del Software.*

17. Estévez, I.P., *MÉTRICAS PARA EL CONTROL DE PROYECTOS DE SOFTWARE*. Junio 2002, INSTITUTO SUPERIOR POLITÉCNICO “JOSÉ ANTONIO ECHEVERRÍA” FACULTAD DE INGENIERÍA INDUSTRIAL: Cuba.

18. Alvarez, J.C.G. *Controles y Métricas Técnicas del Software*. Febrero 2007.

19. *Disciplina Análisis y Diseño.* 2007 [cited; available from: http://teleformacion.uci.cu/file.php/43/Materiales_Basicos/Conferencias/Conferencia_1/Material_de_Apoyo

20. *Flujo de Análisis y Diseño. Modelo de Análisis*. 2007 [cited; Available from: http://teleformacion.uci.cu/mod/resource/view.php?id=10349

# Bibliografía Consultada

Alonso, E. B. (2007). Medición y métricas del software. Universidade de Vigo.

Bastarrica, M. C. (s.f.). Atributos de Calidad y Arquitectura del Software. Universidad de Chile.

Daniele, M. (2007). Métricas de Software. UNRC.

Estrada, A. F. (2003). Aseguramiento de Calidad de Software. La Habana, Cuba.

Estrada, D. A. (2005). Gestión de Proyectos. Métricas de Software. La Habana, Cuba.

Fuente, A. A. (s.f.). NECESIDAD DE SISTEMAS FORMALES DE METRICAS PARA PROYECTOS DE SOFTWARE OO. Universidad de Oviedo .

ISO/IEC 9126 – 3 TECHNICAL REPORT (TR) – Internal metrics (2003)

María N. Moreno García, F. J. (s.f.). Medición de la calidad del software en el ámbito de la especificación de requisitos. Universidad de Salamanca.

Medrano, J. M. (s.f.). Medida de las subcaracterísticas Capacidad de Analisis y Capacidad de Cambio mediante la norma ISO/IEC 9126. INDRA.

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

Navarro, A. (2003). Proceso de software y métricas de proyectos.

PRESSMAN, R. S. *"Ingeniería del Software. Un Enfoque Práctico".* 1998. p.

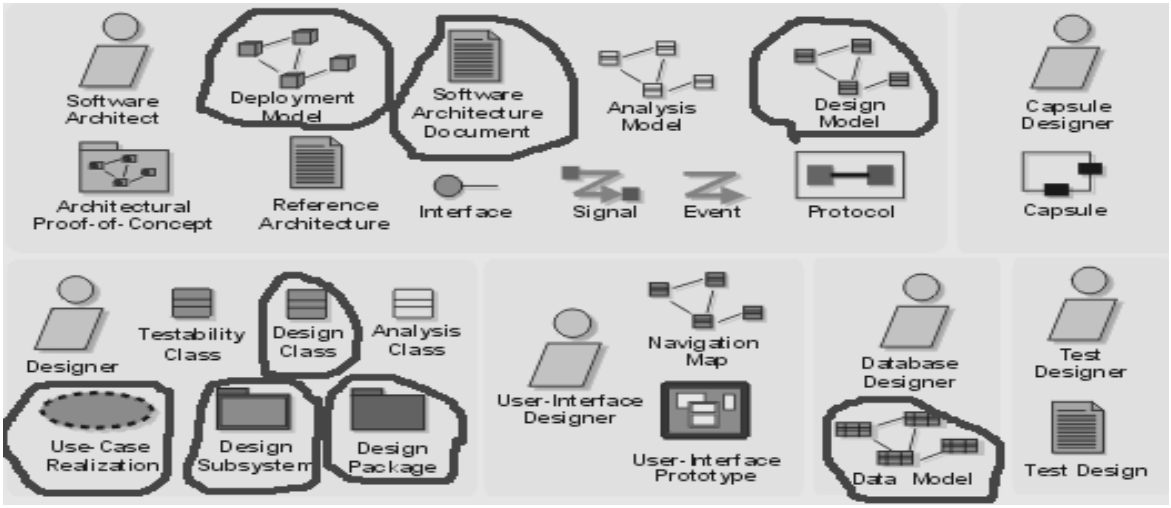
RUP. *“Rational Unified Process”.* 2003.

WIKIPEDIA1. *"The Free Enciclopedia"*, 2006.

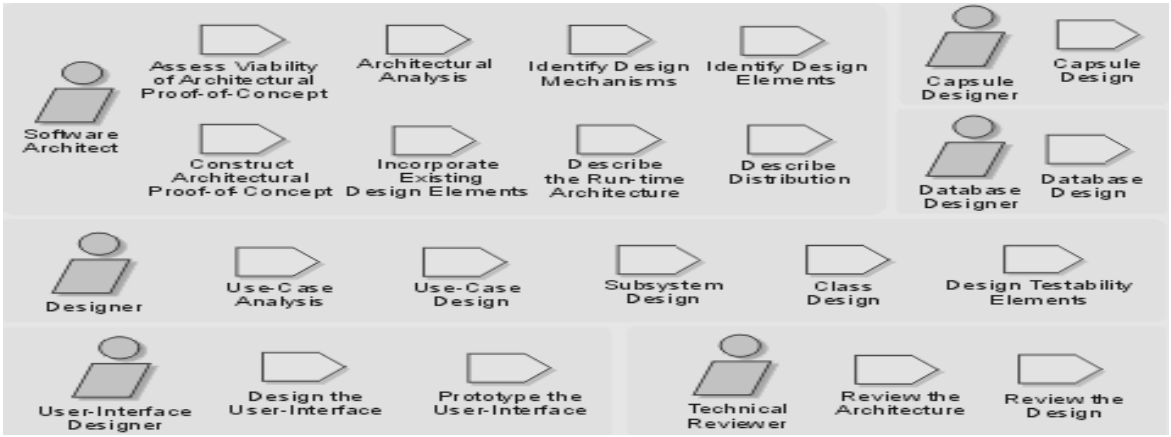
# Anexos

## *5.1 Anexo1- Principales actividades y artefactos de la disciplina de Análisis y Diseño propuesta por RUP.*

Los trabajadores de esta disciplina así como los artefactos que son responsabilidad de cada trabajador, aparecen detallados a continuación; aunque se aconseja que se traten los principales **(estan subrayados).**



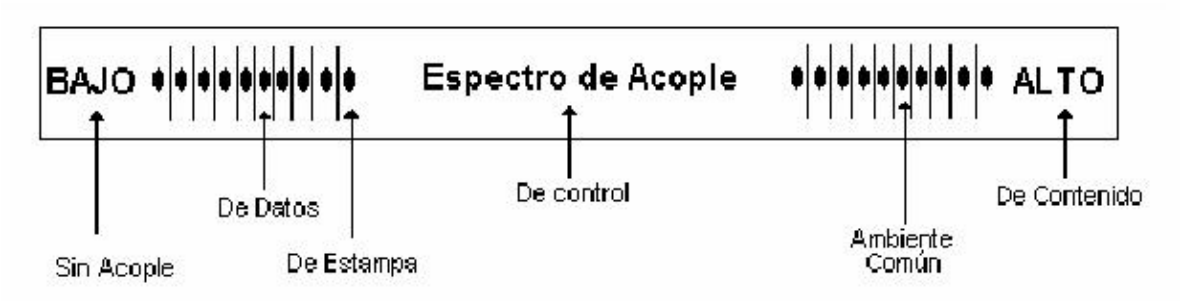

A continuación se muestra el flujo de actividades para Análisis y Diseño que propone RUP y los Trabajadores responsables en cada caso.



## *5.2 Anexo 2 - Características del Diseño*

### Acoplamiento

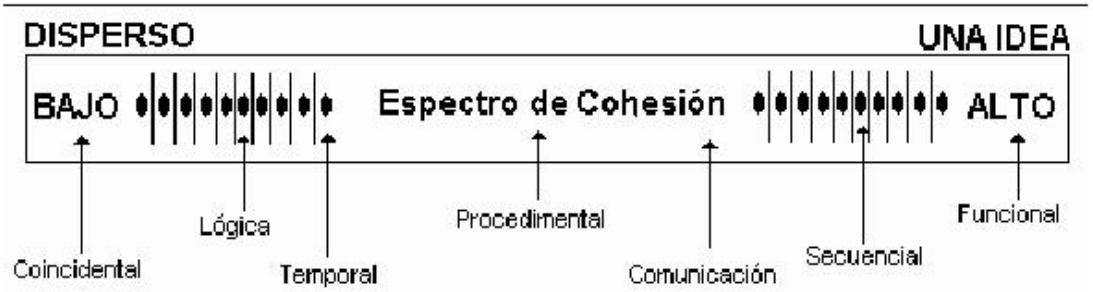
En la siguiente figura se puede apreciar el espectro de acople, donde se identifican bien las variables de acoplamiento.




**Espectro de Acople.**

### Cohesión

En la siguiente figura se ilustra detalladamente el espectro de cohesión.




**Espectro de Cohesión.**

## *5.3 Anexo 3- Relación entre Estabilidad y Abstracción de subsistemas*

| Subsistema/Métrica | A | I | Consecuencias |
|---|---|---|---|
| **Subsistema1** | 0 | 0 | Altamente estable y altamente concreto. Es un subsistema excesivamente rígido. No deseable |
| **Subsistema2** | 1 | 1 | Máxima abstracción y no tiene dependencias. Demasiado rígido porque las abstracciones no han sido extendidas.<br>Quizás imposible. No deseable |
| **Subsistema3** | 0.5 | 0.5 | Parcialmente extensible porque es parcialmente abstracto. Es parcialmente estable por esto las extensiones no son sujeto de máxima inestabilidad.<br>La estabilidad esta balanceada con la abstracción. |

## *5.4 Anexo 4 - 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 _____

Experto # _____

El peso total asignado debe ser 100, usted 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

Grupo No. 1: Criterios de mérito científico

Valor científico de la propuesta.

Peso: _____

Calidad de la investigación.

Peso: _____

Aporte científico.

Peso: _____

Novedad científica.

Peso: _____

Grupo No. 2: Criterios implantación

Satisfacción de las necesidades de los ingenieros de software.

Peso: _____

Necesidad del empleo de la propuesta.

Peso: _____

Uso de la metodología RUP

Peso: _____

Los ingenieros de software deben de tener conocimiento sobre el flujo de trabajo análisis y diseño propuesto por la metodología RUP.

Peso: _____

Los ingenieros de software no deben de tener conocimiento del proceso de medición ni de las métricas para ser aplicadas.

Peso: _____

Grupo No.3: Criterios de flexibilidad

Adaptabilidad a proyectos productivos que utilicen la metodología RUP.

Peso: _____

Uso de las métricas necesarias para retroalimentar el proceso de evaluación de la calidad en los proyectos productivos orientados a objeto.

Peso: _____

Facilidad de entendimiento de las métricas.

Peso: _____

Flexibilidad en la interpretación de los resultados de las métricas aplicadas.

Peso: _____

Grupo No.4: Criterios de impacto

Repercusión en los proyectos productivos que utilicen la metodología RUP.

Peso: _____

Aceptación de la propuesta por los líderes de proyecto.

Peso: _____

Organización en el proceso de aplicación de la métrica.

Peso: _____

Impacto en el área a la cual está destinada.

Peso: _____

## *5.5 Anexo 5 - Guía para la evaluación*

**Modelo No. 2**

**Guía para la evaluación.**

Fecha de recepción _____

Fecha de entrega _____

Experto # _____

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

Grupo No. 1: Criterios de mérito científico

Valor científico de la propuesta.

Peso: _____

Calidad de la investigación.

Peso: _____

Aporte científico.

Peso: _____

Novedad científica.

Peso: _____

Grupo No. 2: Criterios implantación

Satisfacción de las necesidades de los ingenieros de software.

Peso: _____

Necesidad del empleo de la propuesta.

Peso: _____

Uso de la metodología RUP

Peso: _____

Los ingenieros de software deben de tener conocimiento sobre el flujo de trabajo análisis y diseño propuesto por la metodología RUP.

Peso: _____

Los ingenieros de software no deben de tener conocimiento del proceso de medición ni de las métricas para ser aplicadas.

Peso: _____

Grupo No.3: Criterios de flexibilidad

Adaptabilidad a proyectos productivos que utilicen la metodología RUP.

Peso: _____

Uso de las métricas necesarias para retroalimentar el proceso de evaluación de la calidad en los proyectos productivos orientados a objeto.

Peso: _____

Facilidad de entendimiento de las métricas.

Peso: _____

Flexibilidad en la interpretación de los resultados de las métricas aplicadas.

Peso: _____

Grupo No.4: Criterios de impacto

Repercusión en los proyectos productivos que utilicen la metodología RUP.

Peso: _____

Aceptación de la propuesta por los líderes de proyecto.

Peso: _____

Organización en el proceso de aplicación de la métrica.

Peso: _____

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 malos.

___ Malo: No aplicable.

Valoración final

Sugerencias del evaluador para mejorar la calidad del proyecto

Elementos críticos que deben mejorarse.

## *5.6 Anexo 6 - Tabla de los valores del peso relativos a cada criterio*

| G | C/E | $E_1$ | $E_2$ | $E_3$ | $E_4$ | $E_5$ | $E_6$ | $E_7$ | $E_p$ |
|---|---|---|---|---|---|---|---|---|---|
| **25** | **$C_1$** | 5 | 15 | 10 | 4 | 5 | 6 | 8 | **7,571** |
| | **$C_2$** | 10 | 7 | 9 | 17 | 15 | 8 | 10 | **10,857** |
| | **$C_3$** | 5 | 2 | 3 | 4 | 5 | 6 | 4 | **4,142** |
| | **$C_4$** | 5 | 1 | 3 | 0 | 0 | 5 | 3 | **2,428** |
| **30** | **$C_5$** | 8 | 5 | 5 | 6 | 6 | 4 | 4 | **5,428** |
| | **$C_6$** | 10 | 13 | 8 | 10 | 10 | 7 | 6 | **9,142** |
| | **$C_7$** | 5 | 5 | 7 | 8 | 8 | 8 | 8 | **7,000** |
| | **$C_8$** | 5 | 5 | 10 | 6 | 6 | 6 | 8 | **6,571** |
| | **$C_9$** | 2 | 2 | 0 | 0 | 0 | 5 | 4 | **1,857** |
| **20** | **$C_{10}$** | 10 | 5 | 8 | 9 | 8 | 5 | 7 | **7,428** |
| | **$C_{11}$** | 6 | 9 | 8 | 5 | 6 | 4 | 4 | **6,000** |
| | **$C_{12}$** | 2 | 3 | 2 | 3 | 3 | 6 | 4 | **3,285** |
| | **$C_{13}$** | 2 | 3 | 2 | 3 | 3 | 5 | 5 | **3,285** |
| **25** | **$C_{14}$** | 10 | 10 | 6 | 8 | 8 | 10 | 8 | **8,571** |
| | **$C_{15}$** | 5 | 5 | 7 | 2 | 2 | 4 | 5 | **4,285** |
| | **$C_{16}$** | 5 | 5 | 6 | 8 | 8 | 6 | 5 | **6,142** |
| | **$C_{17}$** | 5 | 5 | 6 | 7 | 7 | 5 | 7 | **6,000** |
| **T** | | **100** | **100** | **100** | **100** | **100** | **100** | **100** | **100** |

## *5.7 Anexo 7- Tabla para el cálculo de Concordancia.*

| Expertos/Criterios | $E_1$ | $E_2$ | $E_3$ | $E_4$ | $E_5$ | $E_6$ | $E_7$ | $\sum E$ | $E_p$ | $\Delta C$ | $\Delta C^2$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **$C_1$** | 5 | 15 | 10 | 4 | 5 | 6 | 8 | 53 | 7,571 | 3 | 9 |
| **$C_2$** | 10 | 7 | 9 | 17 | 15 | 8 | 10 | 76 | 10,857 | 26 | 676 |
| **$C_3$** | 5 | 2 | 3 | 4 | 5 | 6 | 4 | 29 | 4,142 | 21 | 441 |
| **$C_4$** | 5 | 1 | 3 | 0 | 0 | 5 | 3 | 17 | 2,428 | 33 | 1089 |
| **$C_5$** | 8 | 5 | 5 | 6 | 6 | 4 | 4 | 38 | 5,428 | 12 | 144 |
| **$C_6$** | 10 | 13 | 8 | 10 | 10 | 7 | 6 | 64 | 9,142 | 14 | 196 |
| **$C_7$** | 5 | 5 | 7 | 8 | 8 | 8 | 8 | 49 | 7,000 | 1 | 1 |
| **$C_8$** | 5 | 5 | 10 | 6 | 6 | 6 | 8 | 46 | 6,571 | 4 | 16 |
| **$C_9$** | 2 | 2 | 0 | 0 | 0 | 5 | 4 | 13 | 1,857 | 37 | 1369 |
| **$C_{10}$** | 10 | 5 | 8 | 9 | 8 | 5 | 7 | 52 | 7,428 | 2 | 4 |
| **$C_{11}$** | 6 | 9 | 8 | 5 | 6 | 4 | 4 | 42 | 6,000 | 8 | 64 |
| **$C_{12}$** | 2 | 3 | 2 | 3 | 3 | 6 | 4 | 23 | 3,285 | 27 | 729 |
| **$C_{13}$** | 2 | 3 | 2 | 3 | 3 | 5 | 5 | 23 | 3,285 | 27 | 729 |
| **$C_{14}$** | 10 | 10 | 6 | 8 | 8 | 10 | 8 | 60 | 8,571 | 10 | 100 |
| **$C_{15}$** | 5 | 5 | 7 | 2 | 2 | 4 | 5 | 30 | 4,285 | 20 | 400 |
| **$C_{16}$** | 5 | 5 | 6 | 8 | 8 | 6 | 5 | 43 | 6,142 | 7 | 49 |
| **$C_{17}$** | 5 | 5 | 6 | 7 | 7 | 5 | 7 | 42 | 6,000 | 8 | 64 |
| **DC** | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 700 | 99,992 | 260 | 6080 |
| **$M\sum E$** | 50 | | | | | | | | | | |
| **W** | 0,2389 | | | | | | | | | | |
| **$X^2$** | 26,7568 | | | | | | | | | | |

## *5.8 Anexo 8 - Tabla de Distribución Chi Cuadrado*

La siguiente tabla es una parte de la tabla de Distribución Chi Cuadrado.

P = Probabilidad de encontrar un valor mayor o igual que el chi cuadrado tabulado,

$\nu$ = Grados de Libertad

| ν/p | 0,001 | 0,0025 | 0,005 | 0,01 | 0,025 | 0,05 | 0,1 | 0,15 | 0,2 | 0,25 | 0,3 | 0,35 | 0,4 | 0,45 | 0,5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 10,8274 | 9,1404 | 7,8794 | 6,6349 | 5,0239 | 3,8415 | 2,7055 | 2,0722 | 1,6424 | 1,3233 | 1,0742 | 0,8735 | 0,7083 | 0,5707 | 0,4549 |
| 2 | 13,8150 | 11,9827 | 10,5965 | 9,2104 | 7,3778 | 5,9915 | 4,6052 | 3,7942 | 3,2189 | 2,7726 | 2,4079 | 2,0996 | 1,8326 | 1,5970 | 1,3863 |
| 3 | 16,2660 | 14,3202 | 12,8381 | 11,3449 | 9,3484 | 7,8147 | 6,2514 | 5,3170 | 4,6416 | 4,1083 | 3,6649 | 3,2831 | 2,9462 | 2,6430 | 2,3660 |
| 4 | 18,4662 | 16,4238 | 14,8602 | 13,2767 | 11,1433 | 9,4877 | 7,7794 | 6,7449 | 5,9886 | 5,3853 | 4,8784 | 4,4377 | 4,0446 | 3,6871 | 3,3567 |
| 5 | 20,5147 | 18,3854 | 16,7496 | 15,0863 | 12,8325 | 11,0705 | 9,2363 | 8,1152 | 7,2893 | 6,6257 | 6,0644 | 5,5731 | 5,1319 | 4,7278 | 4,3515 |
| 6 | 22,4575 | 20,2491 | 18,5475 | 16,8119 | 14,4494 | 12,5916 | 10,6446 | 9,4461 | 8,5581 | 7,8408 | 7,2311 | 6,6948 | 6,2108 | 5,7652 | 5,3481 |
| 7 | 24,3213 | 22,0402 | 20,2777 | 18,4753 | 16,0128 | 14,0671 | 12,0170 | 10,7479 | 9,8032 | 9,0371 | 8,3834 | 7,8061 | 7,2832 | 6,8000 | 6,3458 |
| 8 | 26,1239 | 23,7742 | 21,9549 | 20,0902 | 17,5345 | 15,5073 | 13,3616 | 12,0271 | 11,0301 | 10,2189 | 9,5245 | 8,9094 | 8,3505 | 7,8325 | 7,3441 |
| 9 | 27,8767 | 25,4625 | 23,5893 | 21,6660 | 19,0228 | 16,9190 | 14,6837 | 13,2880 | 12,2421 | 11,3887 | 10,6564 | 10,0060 | 9,4136 | 8,8632 | 8,3428 |
| 10 | 29,5879 | 27,1119 | 25,1881 | 23,2093 | 20,4832 | 18,3070 | 15,9872 | 14,5339 | 13,4420 | 12,5489 | 11,7807 | 11,0971 | 10,4732 | 9,8922 | 9,3418 |
| 11 | 31,2635 | 28,7291 | 26,7569 | 24,7250 | 21,9200 | 19,6752 | 17,2750 | 15,7671 | 14,6314 | 13,7007 | 12,8987 | 12,1836 | 11,5298 | 10,9199 | 10,3410 |
| 12 | 32,9092 | 30,3182 | 28,2997 | 26,2170 | 23,3367 | 21,0261 | 18,5493 | 16,9893 | 15,8120 | 14,8454 | 14,0111 | 13,2661 | 12,5838 | 11,9463 | 11,3403 |
| 13 | 34,5274 | 31,8830 | 29,8193 | 27,6882 | 24,7356 | 22,3620 | 19,8119 | 18,2020 | 16,9848 | 15,9839 | 15,1187 | 14,3451 | 13,6356 | 12,9717 | 12,3398 |
| 14 | 36,1239 | 33,4262 | 31,3194 | 29,1412 | 26,1189 | 23,6848 | 21,0641 | 19,4062 | 18,1508 | 17,1169 | 16,2221 | 15,4209 | 14,6853 | 13,9961 | 13,3393 |
| 15 | 37,6978 | 34,9494 | 32,8015 | 30,5780 | 27,4884 | 24,9958 | 22,3071 | 20,6030 | 19,3107 | 18,2451 | 17,3217 | 16,4940 | 15,7332 | 15,0197 | 14,3389 |
| 16 | 39,2518 | 36,4555 | 34,2671 | 31,9999 | 28,8453 | 26,2962 | 23,5418 | 21,7931 | 20,4651 | 19,3689 | 18,4179 | 17,5646 | 16,7795 | 16,0425 | 15,3385 |
| 17 | 40,7911 | 37,9462 | 35,7184 | 33,4087 | 30,1910 | 27,5871 | 24,7690 | 22,9770 | 21,6146 | 20,4887 | 19,5110 | 18,6330 | 17,8244 | 17,0646 | 16,3382 |
| 18 | 42,3119 | 39,4220 | 37,1564 | 34,8052 | 31,5264 | 28,8693 | 25,9894 | 24,1555 | 22,7595 | 21,6049 | 20,6014 | 19,6993 | 18,8679 | 18,0860 | 17,3379 |
| 19 | 43,8194 | 40,8847 | 38,5821 | 36,1908 | 32,8523 | 30,1435 | 27,2036 | 25,3289 | 23,9004 | 22,7178 | 21,6891 | 20,7638 | 19,9102 | 19,1069 | 18,3376 |
| 20 | 45,3142 | 42,3358 | 39,9969 | 37,5663 | 34,1696 | 31,4104 | 28,4120 | 26,4976 | 25,0375 | 23,8277 | 22,7745 | 21,8265 | 20,9514 | 20,1272 | 19,3374 |
| 21 | 46,7963 | 43,7749 | 41,4009 | 38,9322 | 35,4789 | 32,6706 | 29,6151 | 27,6620 | 26,1711 | 24,9348 | 23,8578 | 22,8876 | 21,9915 | 21,1470 | 20,3372 |
| 22 | 48,2676 | 45,2041 | 42,7957 | 40,2894 | 36,7807 | 33,9245 | 30,8133 | 28,8224 | 27,3015 | 26,0393 | 24,9390 | 23,9473 | 23,0307 | 22,1663 | 21,3370 |
| 23 | 49,7276 | 46,6231 | 44,1814 | 41,6383 | 38,0756 | 35,1725 | 32,0069 | 29,9792 | 28,4288 | 27,1413 | 26,0184 | 25,0055 | 24,0689 | 23,1852 | 22,3369 |
| 24 | 51,1790 | 48,0336 | 45,5584 | 42,9798 | 39,3641 | 36,4150 | 33,1962 | 31,1325 | 29,5533 | 28,2412 | 27,0960 | 26,0625 | 25,1064 | 24,2037 | 23,3367 |
| 25 | 52,6187 | 49,4351 | 46,9280 | 44,3140 | 40,6465 | 37,6525 | 34,3816 | 32,2825 | 30,6752 | 29,3388 | 28,1719 | 27,1183 | 26,1430 | 25,2218 | 24,3366 |
| 26 | 54,0511 | 50,8291 | 48,2898 | 45,6416 | 41,9231 | 38,8851 | 35,5632 | 33,4295 | 31,7946 | 30,4346 | 29,2463 | 28,1730 | 27,1789 | 26,2395 | 25,3365 |
| 27 | 55,4751 | 52,2152 | 49,6450 | 46,9628 | 43,1945 | 40,1133 | 36,7412 | 34,5736 | 32,9117 | 31,5284 | 30,3193 | 29,2266 | 28,2141 | 27,2569 | 26,3363 |
| 28 | 56,8918 | 53,5939 | 50,9936 | 48,2782 | 44,4608 | 41,3372 | 37,9159 | 35,7150 | 34,0266 | 32,6205 | 31,3909 | 30,2791 | 29,2486 | 28,2740 | 27,3362 |
| 29 | 58,3006 | 54,9662 | 52,3355 | 49,5878 | 45,7223 | 42,5569 | 39,0875 | 36,8538 | 35,1394 | 33,7109 | 32,4612 | 31,3308 | 30,2825 | 29,2908 | 28,3361 |

## *5.9 Anexo 9 - Tablas para la calificación de cada criterio*

| Criterios | Clasificación (c) | | | | | P | P x c |
|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | | |
| **C1** | | | | X | | 0,07571 | 0,30284 |
| **C2** | | | | X | | 0,10857 | 0,43428 |
| **C3** | | X | | | | 0,04142 | 0,08284 |
| **C4** | | X | | | | 0,02428 | 0,04856 |
| **C5** | | | X | | | 0,05428 | 0,16284 |
| **C6** | | | | X | | 0,09142 | 0,36568 |
| **C7** | | | | X | | 0,07 | 0,28 |
| **C8** | | | | X | | 0,06571 | 0,26284 |
| **C9** | | X | | | | 0,01857 | 0,03714 |
| **C10** | | | | X | | 0,07428 | 0,29712 |
| **C11** | | | | X | | 0,06 | 0,24 |
| **C12** | | | X | | | 0,03285 | 0,09855 |
| **C13** | | | X | | | 0,03285 | 0,09855 |
| **C14** | | | | | X | 0,08571 | 0,42855 |
| **C15** | | | X | | | 0,04285 | 0,12855 |
| **C16** | | | X | | | 0,06142 | 0,18426 |
| **C17** | | | | X | | 0,06 | 0,24 |
| **Total** | | | | | | | 3,6926 |
| **IA** | **0,73852** | | | | | | |

# Glosario de Términos

**Análisis (flujo de trabajo):** Flujo de trabajo cuyo propósito principal es obtener una comprensión más precisa de los requisitos, refinarlos y estructurarlos, utilizar el lenguaje de los desarrolladores para analizar con profundidad los requisitos funcionales y proporcionar una visión general del sistema.

**Artefactos:** Son los elementos de entrada y salida de las actividades. Son productos tangibles del proyecto. Las cosas que el proyecto produce o usa para componer el producto final (modelos, documentos, código, ejecutables)

**Atributo:** Una unidad con nombre de un clasificador que describe el rango de valores que las instancias de una propiedad pueden tomar.

**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.

**Clase:** Una descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, relaciones y semántica.

**Control de Calidad:** Actividades para evaluar la calidad de los productos desarrollados.

**Componente:** Que forma parte de alguna cosa o de su composición.

**Diseño (flujo de trabajo):** Flujo de trabajo fundamental cuyo propósito principal es el de formular modelos que se centran en los requisitos no funcionales y el dominio de la solución, y que prepara para la implementación y pruebas del sistema.

**Especificación de Requisitos del Software:** Forma de representar los requisitos. Puede ser en un documento o en un gráfico.

**Experto:** Persona que aporta conocimientos o experiencia específica con respecto a una organización, proceso, actividad o materia que se vaya a auditar.

**Evaluación:** La evaluación es el componente más olvidado, entre otras cosas, por la dificultad que implica su definición, pues muchas veces se le confunde con el término de medición. Por evaluación entendemos el establecer un juicio o valor sobre algo. No es sinónimo ni debe confundirse con la medición, pues este término se refiere al proceso de colección de datos, la mayoría de las veces cuantitativos, los cuales luego serán utilizados como base para establecer los juicios. Por lo tanto, el concepto evaluación abarca más que el de medición, pues al medir no necesariamente se está

evaluando. Pero para realizar una adecuada evaluación se necesita que el proceso de medición sea válido y confiable.

**Evaluación cuantitativa:** Procedimientos que requieren medir y cuantificar los fenómenos para describir causas y efectos y explicar relaciones entre variables independientes (tratamiento) y variables dependientes (resultados).

**Evaluación cualitativa:** Cada fenómeno es considerado como algo único que debe ser analizado en su ambiente natural y con la utilización de procedimientos e instrumentos que permitan captarlos en su integridad.

**Evaluación de la calidad**

**Flujo de trabajo (workflow en inglés):** Es el estudio de los aspectos operacionales de una actividad de trabajo: cómo se estructuran las tareas, cómo se realizan, cuál es su orden correlativo, cómo se sincronizan, cómo fluye la información que soporta las tareas y cómo se le hace seguimiento al cumplimiento de las tareas.

**Indicadores:** Métricas que se calculan a partir del resultado de otras métricas calculadas.

**Metodologías de desarrollo de software:** Son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos software.

**Métrica de calidad del software:** Una función cuyas entradas son los datos del software y que la salida es un solo valor numérico que puede interpretarse como el grado en que el software posee un atributo dado que afecta su calidad.
NOTA: Esta definición difiere de la definición de métrica de calidad encontrada en IEEE Std 610.12-1990.

**Métrica del proceso:** Una métrica usada para medir características de los métodos, técnicas, y herramientas empleadas en el desarrollo, implementación, y mantenimiento el sistema del software.

**Métrica del producto:** Una métrica usada para medir las características de cualquier producto intermedio o final del proceso de desarrollo del software.

**Métricas directas:** Métricas que se obtienen a partir de la observación de la manifestación directa de datos.

**Métricas indirectas:** Métricas que se calculan a partir de otras.

**Modelos de Calidad:** En los modelos de calidad, la calidad se define de forma jerárquica. Resuelven la complejidad mediante la descomposición. La calidad es un concepto que se deriva de un conjunto de subconceptos.

**Proceso:** Conjunto de actividades mutuamente relacionadas o que interactúan, las cuales transforman elementos de entrada en resultados, utilizando recursos.

**Producto:** Resultado de cada etapa.

**Proyecto:** Proceso único consistente en un conjunto de actividades coordinadas y controladas con fechas de inicio y de finalización, llevadas a cabo para lograr un objetivo conforme con requisitos específicos.

**Recurso:** Procedimiento o medio del que se dispone para satisfacer una necesidad, llevar a cabo una tarea o conseguir algo. Pueden ser personas o recursos materiales.

**Requisito:** Condición necesaria para que algo se cumpla.

**Riesgo (software):** Proximidad a un daño o peligro.

**Sistema:** Conjunto de elementos mutuamente relacionados o que actúan entre sí.

**Software**: Son las instrucciones electrónicas que van a indicar al ordenador que es lo que tiene que hacer. También se puede decir que son los programas usados para dirigir las funciones de un sistema de computación o un hardware.

**Validación:** Confirmación mediante el suministro de evidencia objetiva de que se han cumplido los requisitos para una utilización o aplicación especifica prevista.

**Validación de los requisitos:** Actividad para detectar omisiones o ambigüedades en los requisitos.

**Variable:** Magnitud que puede tener un valor cualquiera de los comprendidos en un conjunto.