CAPÍTULO II

2. MARCO TEÓRICO

Este capítulo engloba el marco teórico del análisis que se efectúa, define el proceso de desarrollo del software y todo lo referente a él para un mejor entendimiento. Además se define y explica acerca de las métricas y por último se describe detalladamente la técnica estadística que se utilizó para llevar a cabo el análisis de los datos.

2.1. La Ingeniería del Software[16]

La Ingeniería de software es la rama de la ingeniería que crea y mantiene las aplicaciones de software aplicando tecnologías y prácticas de las ciencias computacionales, manejo de proyectos, ingeniería, el ámbito de la aplicación, y otros campos.

Software es el conjunto de instrucciones que permite al hardware de la computadora desempeñar trabajo útil. En las últimas décadas del siglo XX, las reducciones de costo en hardware llevaron a que el software fuera un componente ubicuo de los dispositivos usados por las sociedades industrializadas.

[16] Ingeniería del Software, disponible en http://es.wikipedia.org/wiki/Ingenier%C3%ADa_de_software

La ingeniería de software, como las disciplinas tradicionales de ingeniería, tiene que ver con el costo y la confiabilidad. Algunas aplicaciones de software contienen millones de líneas de código que se espera que se desempeñen bien en condiciones siempre cambiantes.

El término ingeniería de software se usa con una variedad de significados diferentes:

- Como el término usual contemporáneo de un amplio rango de actividades que se solía llamar programación y análisis de sistemas;
- Como un término amplio de todos los aspectos de la práctica de la programación de computadoras, en oposición a la teoría, que es llamada ciencia computacional o computación.

La ingeniería de software es "la aplicación de un método sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento del software".

2.2. El Proceso de Desarrollo del Software[12]

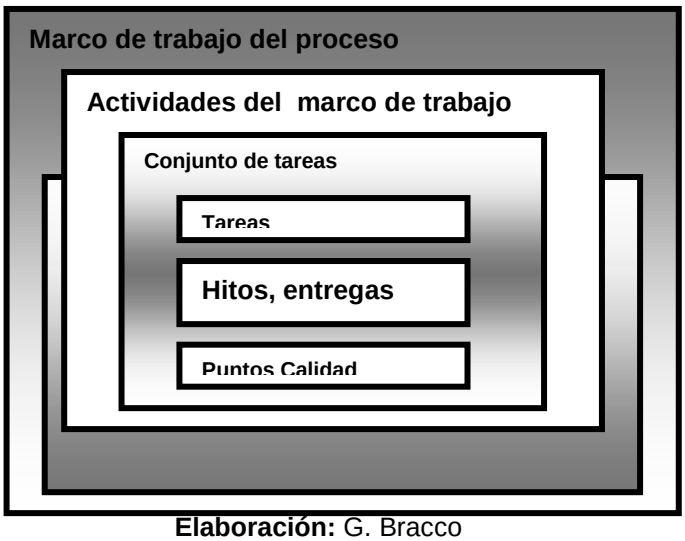
Se define como un conjunto estructurado de actividades requeridas para desarrollar un sistema de software, por ejemplo: Especificación, Diseño, Validación, Evolución, etc. Las actividades varían dependiendo de la

[12] Pressman, Roger S., 1998, Ingeniería del Software: un enfoque práctico, 4ª Edición, Mc. Graw Hill de Interamericana de España

organización y del tipo de sistema a desarrollarse, estas deben estar explícitamente modeladas si va a ser bien administrado.

Un proceso de software se puede caracterizar, como se muestra en la Figura 2.1, un marco común del proceso, definiendo un pequeño número de actividades del marco de trabajo que son aplicables a todos los proyectos del SW, con independencia a su tamaño o complejidad. Un conjunto de tareas, cada una en su colección de tareas de ingeniería del software, hitos del proyecto, posibles entregas y productos de trabajo del software, y puntos de garantía de calidad, que permiten que las actividades del marco de trabajo se adapten a las características del proyecto del software y a los requisitos del equipo del proyecto. Las actividades de protección son independientes de cualquier actividad del marco de trabajo y aparecen durante todo el proceso

Figura 2.1
Proceso desarrollo del software



Elaboración: G. Bracco

2.2.1. Principales actividades

Aunque existen muchos procesos diferentes de software, tienen actividades fundamentales que son comunes para todos ellos. Estas son:

- Especificación del software.
- Diseño e implementación del software
- Validación del software
- Evolución del software

2.2.1.1. Especificación del software

Sirve para establecer que servicios se requieren del sistema y las restricciones de operación del mismo. Esta actividad a menudo se llama ingeniería de requerimientos. Se trata de especificar las funciones que debe desempeñar el software, las exigencias que debe satisfacer en el desempeño de esas funciones y las que deben cumplir los procesos de producción.

Existen cuatro fases principales en el proceso de ingeniería de requerimientos:

- Estudio de factibilidad. Se estima si las necesidades del usuario se pueden satisfacer con las tecnologías actuales de software y hardware.

- Obtención y análisis de requerimientos. Este es el proceso de derivar los requerimientos del sistema por medio de la observación de los sistemas existentes, discusiones con los potenciales usuarios potenciales y proveedores.
- Especificación de requerimientos. Esta es la actividad de traducir la información recolectada
- Validación de requerimientos. Esta actividad comprueba la veracidad, consistencia y compleción de los requerimientos.

El proceso de reunión de requisitos se centra especialmente en el software. Para comprender la naturaleza de los programas a construirse, el ingeniero del software debe comprender el dominio de información del software, así como la función requerida, comportamiento, rendimiento, e interconexión. El cliente documenta y repasa los requisitos del sistema y del software.

2.2.1.2. Diseño e implementación del software

El diseño consiste en la búsqueda y especificación (Especificación del Software) de una estructura para el Software que satisfaciendo los requerimientos se pueda construir con los recursos disponibles. Es una descripción de la estructura del software que se va a

implementar, los datos, las interfaces y (a veces) los algoritmos utilizados.

Las actividades del proceso de diseño son:

- Diseño arquitectónico. Los subsistemas conforman el sistema y su relación se identifica y documenta.
- Especificación abstracta. Para cada subsistema se produce una especificación abstracta de sus servicios y las restricciones bajo las cuales opera.
- Diseño de la interfaz. Para cada subsistema se diseña y documenta su interacción con otros subsistemas.
- Diseño de componentes. Se asignan servicios a los diferentes componentes y se diseñan sus interfaces.
- Diseño de la estructura de datos. Se diseña en detalle y especifica la estructura de datos a utilizarse en la implementación del sistema.
- Diseño de algoritmos. Se diseñan en detalle y especifican los algoritmos utilizados para proveer los servicios.

El diseño del software es realmente un proceso de muchos pasos que se centra en cuatro atributos distintos de un programa: estructura de datos, arquitectura de software, representaciones de interfaz y detalle procedimental (algoritmo). El proceso de diseño

traduce requisitos en una representación del software que se pueda evaluar por calidad antes de que comience la generación del código.

El paso de generación de código traduce el diseño a una forma legible por la máquina. Si se lleva a cabo el diseño de una forma detallada, la generación de código se realiza mecánicamente.

La implementación comprende la elaboración del código de programa y la creación de las estructuras de datos persistentes. Es el proceso de convertir una especificación del sistema en un sistema ejecutable. Incluye los procesos de diseño y programación de software.

2.2.1.3. Validación del software

La validación tiene como objetivo garantizar que los productos software satisfacen sus requerimientos. Se utiliza para mostrar que el sistema está acorde a su especificación y cumple con las expectativas del usuario que lo comprará. Las etapas en el proceso de prueba son:

- Prueba de unidades. Se prueban los componentes individuales para asegurarse de que operan correctamente.

- Prueba de módulos. Un módulo es una colección de componentes dependientes, como una clase de objetos, se prueban procedimientos y funciones.
- Prueba de subsistemas. Esta fase comprende la colección de pruebas de los módulos que integran el subsistema.
- Prueba del sistema. Este proceso comprende encontrar errores que son el resultado de interacciones no previstas entre los subsistemas y su interfaz. Se valida también que el sistema cumpla sus requerimientos funcionales y no funcionales y probar las propiedades emergentes del sistema.
- Prueba de aceptación. Esta es la etapa final en el proceso de pruebas antes de que se acepte que el sistema se ponga en operación. Este se prueba con los datos proporcionados por el cliente más que con datos de entrada simulados.

El proceso de validación y pruebas se centra en los procesos lógicos internos del software, asegurando que todas las sentencias se han comprobado, y en los procesos externos funcionales, es decir, la realización de las pruebas para la detección de errores y el sentirse seguro de que la entrada definida produzca resultados reales de acuerdo con los resultados requeridos.

2.2.1.4. Evolución del software

Es el proceso de cambio del sistema una vez que se ha puesto en funcionamiento. Aunque los costos de “mantenimiento” son a menudo varias veces más que los costos iniciales de desarrollo, el proceso de mantenimiento se considera menos problemático que el desarrollo del software original. Es más realista considerar a la ingeniería de software como un proceso evolutivo en el cual el software se cambia continuamente durante su período de vida como respuesta a los requerimientos cambiantes y necesidades del usuario.

2.2.2. Modelos del proceso del software

Cada modelo de proceso representa un proceso desde una perspectiva por lo que sólo provee información parcial acerca de ese proceso.

Entre los modelos de procesos más comunes tenemos:

- El modelo de cascada.
- Desarrollo evolutivo
- Desarrollo formal de sistemas
- Desarrollo basado en la reutilización.

Los procesos que se basan en cascada y en desarrollo evolutivo se utilizan ampliamente en el desarrollo de sistemas prácticos. La reutilización informal es común en muchos procesos, pero la mayoría de las organizaciones no orientan el desarrollo de software alrededor de esta. Pero esa situación cambia poco a poco, ya que construir software a partir de componentes reutilizables es fundamental para un desarrollo rápido de software.

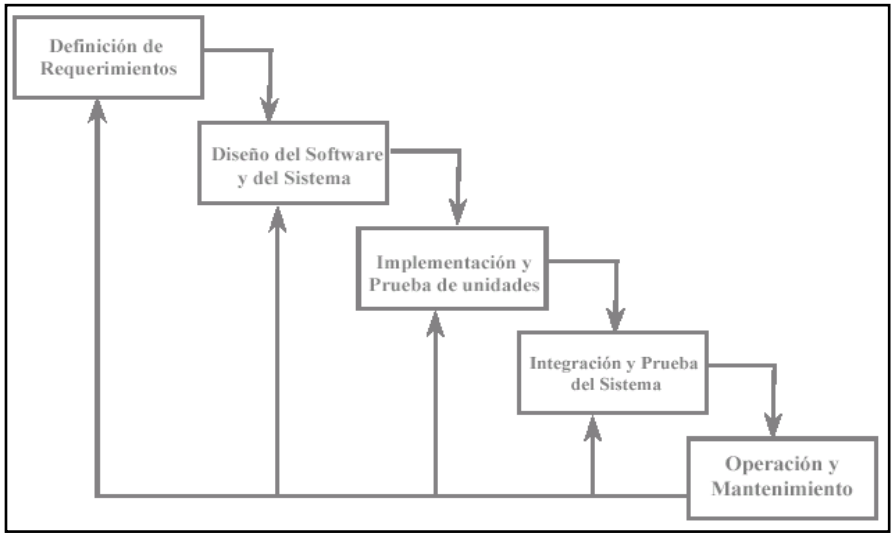
2.2.2.1. Modelo cascada

El primer modelo de proceso de desarrollo de software que se publicó se derivó de otros procesos de ingeniería. Su nombre se debe a la forma de cascada de una fase a otra. Las principales etapas de este modelo se transforman en actividades fundamentales de desarrollo:

- Análisis y definición de requerimientos: los servicios, restricciones y metas del sistema se definen a partir de las consultas con los usuarios.
- Diseño de sistemas y de software. El proceso de diseño de sistemas divide los requerimientos en sistemas de hardware o de software. Establece una arquitectura completa del sistema.

- Implementación y prueba de unidades. Durante esta etapa, el diseño de software se lleva a cabo como un conjunto o unidades de programas.
- Integración y prueba del sistema. Los programas o las unidades de programas se integran y prueban como un sistema complejo para asegurar que se cumplan los requerimientos del software.
- Operación y mantenimiento. Por lo general esta es la fase más larga del ciclo de vida. El mantenimiento implica corregir errores no descubiertos en las etapas anteriores del ciclo de vida.

Figura 2.2
Modelo Cascada



Elaboración: G. Bracco

Problemas con el modelo cascada:

- Los proyectos reales raras veces siguen el modelo secuencial que propone el modelo.
- Es difícil que el cliente exponga explícitamente todos los requerimientos.
- El cliente debe tener paciencia, una versión de trabajo de los programas no estará disponible hasta que el proyecto esté muy avanzado.
- Los responsables del desarrollo del software siempre se retrasan innecesariamente.

2.2.2.2. Desarrollo evolutivo

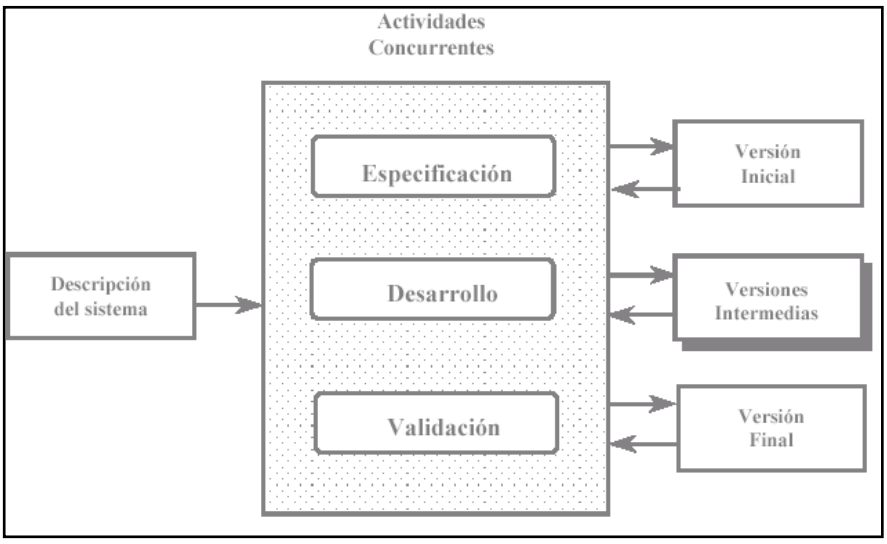
Este se basa en la idea de desarrollar una implementación inicial, exponiéndola a los comentarios del usuario y refinándola a través de las diferentes versiones hasta que se desarrolla un sistema adecuado.

- Desarrollo exploratorio. El objetivo del proceso es trabajar con el cliente para explorar sus requerimientos y entregar un sistema final. El desarrollo empieza con las partes del sistema que se comprenden mejor. El sistema evoluciona agregando nuevos atributos acordes con las propuestas del cliente.

- Prototipos desechables. El objetivo del proceso de desarrollo evolutivo es comprender los requerimientos del cliente y entonces desarrollar una definición mejorada de los requerimientos para el sistema. El prototipo se centra en experimentar con aquellas partes de los requerimientos del cliente que no se comprenden del todo.

La ventaja de un proceso del software que se basa en un enfoque evolutivo es que la especificación se puede desarrollar de forma creciente. Tan pronto como los usuarios desarrollen un mejor entendimiento de su problema.

Figura 2.3
Modelo Desarrollo Evolutivo



Elaboración: G. Bracco

Desventajas de un proceso evolutivo:

- El proceso no es visible. Los administradores tienen que hacer entregas regulares para medir el progreso. Si los sistemas se desarrollan rápidamente, es muy costoso producir documentos que reflejen cada versión del sistema.
- A menudo los sistemas tienen una estructura deficiente. Los cambios continuos tienden a corromper la estructura del software. Incorporar cambios en él se convierte en una tarea difícil y costosa.
- Se requieren herramientas y técnicas especiales. Éstas permiten un desarrollo rápido pero son incompatibles con otras herramientas o técnicas y pocas personas tienen las habilidades necesarias para utilizarlas.

2.2.2.3. Desarrollo formal de sistemas

La especificación de requerimientos de software se refina en una especificación formal detallada que se expresa en notación matemática.

Los procesos de desarrollo de diseño, implementación y pruebas de unidades se reemplazan con un proceso de desarrollo por transformaciones donde la especificación formal se refina, a través de una serie de transformaciones, hasta llegar a un programa.

2.2.2.4. Desarrollo orientado a la reutilización

En la mayoría de los proyectos de software existe algo de reutilización de software. Por lo general, esto pasa cuando las personas que trabajan en el proyecto conocen diseños o código similares al requerido.

Esta reutilización informal es independiente del proceso genérico que se utilice. Sin embargo, en años recientes, ha surgido un enfoque de desarrollo de software (ingeniería de software basada en componentes) que se basa en la reutilización, el cual se está utilizando de forma amplia.

Aunque la etapa de especificación de requerimientos y la de validación son comparables con otros procesos, las etapas intermedias en el proceso orientado a la reutilización son diferentes.

- Análisis de componentes. Dada la especificación de requerimientos, se buscan los componentes para implementar esta especificación.
- Modificación de requerimientos. En esta etapa, los requerimientos se analizan utilizando información acerca de los componentes que se van descubriendo. Entonces estos

componentes se modifican para reflejar componentes disponibles.

- Diseño de sistemas con reutilización. En esta fase se diseña o se reutiliza un marco de trabajo para el sistema. Los diseñadores toman en cuenta los componentes que se reutilizan y organizan el marco de trabajo para que se ajuste a ellos.
- Desarrollo e integración. Para crear el sistema, el software que no se puede comprar se desarrolla, y los componentes y los sistemas comerciales se integran.

2.2.3. Iteración de procesos

Para sistemas muy grandes, existe la necesidad de utilizar diferentes enfoques para las distintas partes del sistema. Los sistemas grandes están hechos usualmente de varios subsistemas no es necesario utilizar el mismo modelo de proceso para todos los subsistemas.

- El desarrollo incremental en el que especificación, diseño e implementación del software se dividen en una serie de incrementos los cuales se desarrollan uno a uno.

- El desarrollo en espiral en el que el desarrollo gira en espiral hacia fuera, empezando con un esbozo inicial y terminando con el desarrollo final del sistema.

La esencia de los procesos iterativos es que la especificación se desarrolla junto con el Software.

2.2.3.1. Desarrollo incremental

El modelo de desarrollo en cascada requiere que los clientes de un sistema cumplan un conjunto de requerimientos antes de que se inicie el diseño, y que el diseñador cumpla estrategias particulares de diseño antes de la implementación. Los cambios de requerimientos durante el desarrollo implican rehacer el trabajo de captura de éstos, de diseño e implementación.

En un proceso de desarrollo incremental, los clientes identifican, de forma superficial los servicios que proveerá el sistema. El proceso de desarrollo incremental tiene varias ventajas:

- Los clientes no tienen que esperar hasta que el sistema completo se entregue para sacar provecho de él. El primer

incremento satisface los requerimientos más críticos de tal forma que el software está disponible para su uso inmediato.

- Los clientes pueden utilizar los incrementos iniciales como un prototipo para obtener experiencia sobre los requerimientos de los incrementos para su uso inmediato.
- Existe un bajo riesgo de fallar en el proyecto total.
- Puesto que los servicios de alta prioridad se entregan primero y los incrementos posteriores se integran a ellos.

2.2.3.2. Desarrollo en espiral

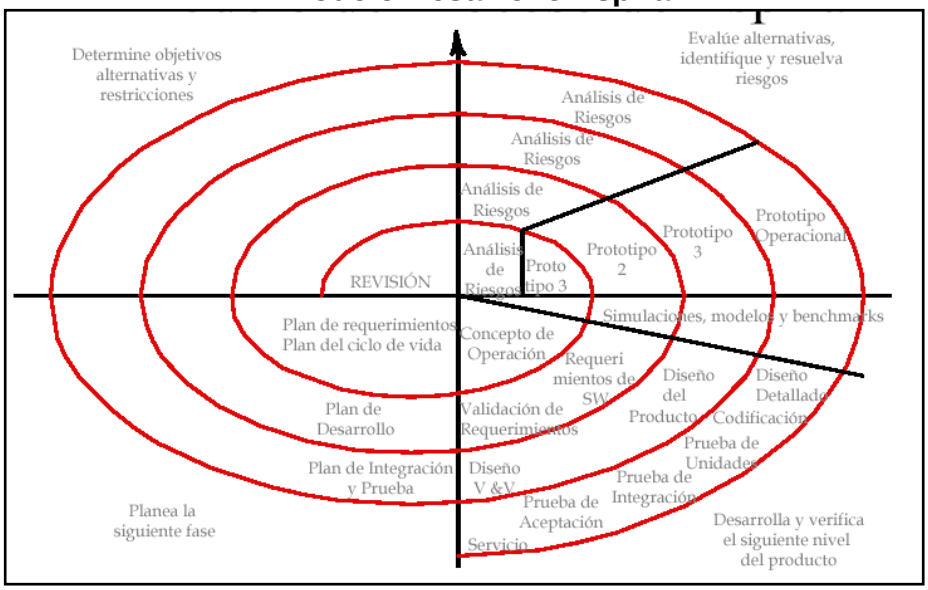
El Modelo Espiral mejora el Modelo de Cascada enfatizando la naturaleza iterativa del proceso de diseño. Eso introduce un ciclo de prototipo iterativo. En cada iteración, las nuevas expresiones que son obtenidas transformando otras dadas son examinadas para ver si representan progresos hacia el objetivo.

2.2.3.2.1. Fases del Modelo Espiral.

- Planteamiento de Objetivos: Se identifican los objetivos específicos para cada fase del proyecto.
- Identificación y reducción de riesgos.: Los riesgos clave se identifican y analizan, y la información sirve para minimizar los riesgos.

- Desarrollo y Validación.: Se elige un modelo apropiado para la siguiente fase del desarrollo.
- Planeación.: Se revisa el proyecto y se trazan planes para la siguiente ronda del espiral.

Figura 2.4
Modelo Desarrollo Espiral



Elaboración: G. Bracco

Ventajas:

- Centra su atención en la reutilización de componentes y eliminación de errores en información descubierta en fases iniciales.
- Los objetivos de calidad son el primer objetivo.
- Integra desarrollo con mantenimiento.

  Provee un marco de desarrollo de hardware/software.

Desventajas:

- El desarrollo contractual especifica el modelo del proceso y los resultados a entregar por adelantado.
- Requiere de experiencia en la identificación de riesgos.

  Requiere refinamiento para uso generalizado.

2.2.4. Requerimientos Funcionales y no Funcionales

A menudo, los requerimientos de sistemas de software se clasifican en funcionales y no funcionales, o como requerimientos del dominio:

2.2.4.1. Requerimientos funcionales

Son declaraciones de los servicios que proveerá el sistema, de la manera en que éste reaccionará a entradas particulares y de cómo se comportará en situaciones particulares. En algunos casos, los requerimientos funcionales de los sistemas también declaran explícitamente lo que el sistema no debe hacer.

2.2.4.2. Requerimientos no funcionales

Son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo, estándares, etc.

2.2.4.3. Requerimientos del dominio

Son requerimientos que provienen del dominio de aplicación del sistema y que reflejan las características de ese dominio. Estos pueden ser funcionales o no funcionales.

2.3. Conceptos básicas de métricas[7]

Según varias definiciones la palabra *métrica* esta muy asociada a al palabra medición o medida, aunque estas tres son distintas. La *medición "*es el proceso por el cual los números o símbolos son asignados a atributos o entidades en el mundo real tal como son descritos de acuerdo a reglas claramente definidas" [Fenton ´91]. Una *medida* "proporciona una indicación cuantitativa de extensión, cantidad, dimensiones, capacidad y tamaño de algunos atributos de un proceso o producto" [Pressman´98]. El IEEE "*Standard Glossary of Software Engineering*" define como *métrica* "una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado*".*

2.3.1. Definición de métricas de software

Michael ['99] define las 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

[7] González Doria, Heidi, 2001, Las métricas de software y su uso en la región, Universidad de las Américas – Puebla

relevante a tiempo, así el administrador junto con el empleo de estás técnicas mejorará el proceso y sus productos". Las métricas de software proveen la información necesaria para la toma de decisiones técnicas.

Un ingeniero del software recopila medidas y desarrolla métricas para obtener indicadores. Un indicador es una métrica o una combinación de métricas que proporcionan una visión profunda del proceso del software, del proyecto del software, o del producto en si. Un indicador permite al gestor de proyectos o a los ingenieros del software ajustar el proceso, el proyecto, o el producto para que las cosas salgan mejor.

La medición es esencial, si es que se desea realmente conseguir la *calidad en software*. Es por eso que existen distintos tipos de métricas para poder evaluar, mejorar y clasificar al software final, en donde serán manejadas dependiendo del entorno de desarrollo del software al cual pretendan orientarse.

2.3.2. Clasificación de Métricas

La clasificación de una métrica de software refleja o describe la conducta del software. A continuación se muestra una breve clasificación de métricas de Software:

- *Métricas de complejidad*: Son todas las métricas de software que definen de una u otra forma la medición de la complejidad;

Tales como volumen, tamaño, anidaciones, costo (estimación), agregación, configuración, y flujo. Estas son los puntos críticos de la concepción, viabilidad, análisis, y diseño de software.

- *Métricas de calidad*: Son todas las métricas de software que definen de una u otra forma la calidad del software; Tales como exactitud, estructuración o modularidad, pruebas, mantenimiento, reusabilidad, cohesión del módulo, acoplamiento del módulo, etc. Estas son los puntos críticos en el diseño, codificación, pruebas y mantenimiento.

- *Métricas de competencia*: Son todas las métricas que intentan valorar o medir las actividades de productividad de los programadores o practicantes con respecto a su certeza, rapidez, eficiencia y competencia. No se ha alcanzado mucho en esta área, a pesar de la intensa investigación académica.

- *Métricas de desempeño*: Corresponden a las métricas que miden la conducta de módulos y sistemas de un software, bajo la supervisión del sistema operativo o hardware. Generalmente tienen que ver con la eficiencia de ejecución, tiempo,

almacenamiento, complejidad de algoritmos computacionales, etc.

Estas clasificaciones de métricas fortalecen la idea de que más de una métrica puede ser deseable para valorar la complejidad y la calidad del software.

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

Del mismo modo existen diferentes tipos de métricas para poder evaluar, mejorar y clasificar al software desde sus inicios hasta el producto final.

2.3.3. Funciones de las métricas del software

Se sabe que las métricas de software pueden desempeñar una de las cuatro siguientes funciones:

- Las métricas pueden ayudarnos a *entender* más acerca de nuestros productos, procesos y servicios de software.
- Las métricas pueden ser empleadas para *evaluar* el software de nuestros productos, procesos y servicios con respecto a los estándares y metas establecidas.
- Las métricas pueden proveer la información que nosotros necesitamos para *controlar* recursos y procesos utilizados en la producción de nuestro software.
- Las métricas pueden ser usadas para *predecir* los atributos de las entidades de software en el futuro.

2.4. Análisis Univariado

El análisis univariado en un sentido amplio, se refiere a todos los métodos estadísticos que analizan la distribución de una sola variable.

Para el análisis en este estudio se considerarán las siguientes medidas descriptivas:

De posición

- Cuartiles
- Mínimo
- Máximo

Centralización o tendencia central

- Media
- Mediana

Dispersión

- Desviación estándar
- Varianza

Forma

- Sesgo o asimetría
- Curtosis o puntiagudez

Los gráficos que se considerarán en el estudio son los Histogramas.

2.5. Tipo de muestreo[11]

Los tipos de muestreo se clasifican en: Probabilístico y No Probabilístico.

Entre los Probabilisticos tenemos:

[11] Pérez, César; 2000, Técnicas de Muestreo Estadístico: Teoría, práctica y aplicaciones informáticas; Editorial Alfaomega; Madrid –España

- Muestreo aleatorio simple
- Muestreo estratificado
- Muestreo sistemático
- Muestreo de conglomerado

Entre los No Probabilistico tenemos:

- Muestreo Intencional
- Muestreo por criterio
- Muestreo por cuotas
- Muestreo sin norma
- Muestreo semiprobabilístico

2.5.1. Obtención de la Muestra

Para la obtención de la muestra se utilizó el muestreo no probabilístico, denominado muestreo sin norma.

Este tipo de muestreo consiste en seleccionar la muestra sujeta a una de las siguientes razones: comodidad, disponibilidad de los informantes o fácil acceso a los datos. Resulta más óptimo aplicar este tipo muestreo en poblaciones homogéneas, pues la población que se estudia es homogénea excepto por pocos casos en que se registran datos aberrantes.

2.6. Análisis Multivariado[8]

En este capitulo se realizará el análisis multivariado para conocer el comportamiento de dos o más variables simultáneamente, con el fin de determinar la forma en que una variable se enlaza con otra y medir el grado de asociación entre ellas, para lo cual aplicaremos las técnicas estadísticas multivariadas tales como: Tablas de Contingencia y Análisis de Homogeneidad.

2.6.1. Análisis de contingencia

En esta sección se presenta un contraste de hipótesis en el que se comprueba la independencia entre dos variables. El análisis es presentado por medio de tablas de contingencia, que son un arreglo bidimensional en el que se detalla los factores a ser analizados con igual o diferentes niveles de información.

Sea A el primer factor con *r* niveles de información y B el segundo factor con *c* niveles de información, se define el modelo de tabla de contingencia:

[8] Jonson, D, 1998, Métodos Multivariados: Aplicados al análisis de los datos, Internacional Thompson Editors, México

Figura 2.5
Tabla de Contingencia

FACTOR A / FACTOR B	NIVEL 1	NIVEL 2	…	NIVEL C	X_i
Nivel 1	X_{11} X_{11}	X_{12} X_{12}	…	X_{1c} X_{1c}	X_1
Nivel 2	X_{21} X_{21}	X_{22} X_{22}	…	X_{1c} X_{1c}	X_2
⋮	⋮	⋮	⋮	⋮	⋮
Nivel r	X_{r1} X_{r1}	X_{r2} X_{r2}	…	X_{rc} X_{rc}	$X_{r.}$
$X_{.j}$	$X_{.1}$	$X_{.2}$	…	$X_{.c}$	$X_{..}$

Elaboración: G. Bracco

Donde:

X_{ij} es el número de valores observados que poseen simultáneamente la i-ésima característica del factor A y la j-ésima característica del factor B.

E_{ij} es el número de observaciones esperadas con la i-ésima característica del factor A y la j-ésima característica del factor B, si H_0 es verdadera y se lo obtiene de la siguiente manera:

$$E_{ij} = \frac{X_{i.} * X_{.j}}{n} = \frac{\sum_{i=1}^{r} X_{ij} * \sum_{j=1}^{c} X_{ij}}{n}$$

$X_{i.}$ es el número de observaciones que poseen la característica i-ésima del factor B

$X_{.j}$ es el número de observaciones que poseen la característica j-ésima del factor A.

$X_{..}$ es el número total de observaciones o *n*.

Luego de obtener la Tabla de Contingencia se realiza el siguiente contraste de hipótesis:

H_0 : Los factores A y B son independientes

Vs.

H_1 : No es verdad H_0

Se puede probar que el estadístico: $\chi^2 = \sum_{i=1}^{h}\sum_{j=1}^{k} (X_{ij} - E_{ij})^2 / E_{ij}$ tiene una distribución Ji – cuadrado con (r-1)*(c-1) grados de libertad, por lo que se rechaza la hipótesis nula a favor de la hipótesis alternativa con $(1-\alpha)100\%$ de confianza si $\chi^2 > \chi^2_{\alpha}(r-1)(c-1)$.

2.6.2. Análisis de Homogeneidad (Homals)

El análisis de homogeneidad cuantifica los datos (categóricos) nominales mediante la asignación de valores numéricos a los casos (los objetos) y a las categorías. El análisis de homogeneidad se conoce también por el acrónimo HOMALS, del inglés homogeneity análisis by means of alternating least squares (análisis de homogeneidad mediante mínimos cuadrados alternantes).

El objetivo de HOMALS es describir las relaciones entre dos o más variables nominales en un espacio de pocas dimensiones que contiene las categorías de las variables así como los objetos pertenecientes a dichas categorías. Los objetos pertenecientes a la misma categoría se representan cerca los unos de los otros, mientras que los objetos de diferentes categorías se representan alejados los unos de los otros. Cada objeto se encuentra lo más cerca posible de los puntos de categoría para las categorías a las que pertenece dicho objeto.

El análisis de homogeneidad es similar al análisis de correspondencias, pero no está limitado a dos variables. Es por ello que el análisis de homogeneidad se conoce también como el análisis de correspondencias múltiples. También se puede ver el análisis de

homogeneidad como un análisis de componentes principales para datos nominales.

El análisis de homogeneidad es mas adecuado que el análisis de componentes principales típico cuando puede que no se conserven las relaciones lineales entre las variables, o cuando las variables se miden a nivel nominal. Además, la interpretación del resultado es mucho más sencilla en HOMALS que en otras técnicas categóricas, como pueden ser las tablas de contingencia.

Estadísticos y gráficos. Los estadísticos que se obtienen del análisis de Homogeneidad son las frecuencias, autovalores, historial de iteraciones, puntuaciones de objeto, cuantificaciones de categoría, medidas de discriminación, gráficos de las puntuaciones de objeto, gráfico de las cuantificaciones de categoría, gráfico de las medidas de discriminación.

2.7. Determinación de los índices de satisfacción

Un índice de satisfacción es una medida del grado en que el desempeño general de un producto o servicio satisface las necesidades y expectativas del cliente. La determinación de índices de satisfacción es cada día más importante para las empresas, porque permiten cuantificar

la calidad del producto o servicio que se ofrece y como éste es percibido por los clientes. Además, tales índices son útiles para comparar los resultados obtenidos con periodos futuros de tiempo.

2.7.1. Metodología

Se aplica el cuestionario al personal de trabajo para conocer la conformidad con las herramientas utilizadas para el desarrollo del software de diversos atributos que a estas le componen.

Los atributos a evaluar en el Sistema Operativo, Plataforma de trabajo, Lenguaje de Programación, Base de Datos y Servidor HTML/WEB son los siguientes:

- Funcionalidad
- Facilidad de Uso
- Confiabilidad
- Rendimiento
- Capacidad de Soporte

Las respuestas obtenidas por la aplicación del cuestionario al personal de trabajo, son utilizados para calcular los índices de satisfacción.

Luego, se determina para cada uno de los atributos que componen las herramientas de trabajo, el número de respuestas del personal para cada valor de la escala de la calificación de la siguiente manera, donde 1 es total desacuerdo y 5 total acuerdo:

$N5_i$ = número de respuestas con escala 5 en el atributo i

$N4_i$ = número de respuestas con escala 4 en el atributo i

$N3_i$ = número de respuestas con escala 3 en el atributo i

$N2_i$ = número de respuestas con escala 2 en el atributo i

$N1_i$ = número de respuestas con escala 1 en el atributo i

Donde:

i representa el atributo i de las herramientas de gestión de trabajo.

Luego se obtiene para cada atributo i de las herramientas, el promedio de la calificación dada por el personal de trabajo.

$$C_i = ((N5_i * 5) + (N4_i * 4) + (N3_i * 3) + (N2_i * 2) + (N1_1 * 1)) / n_i$$

Donde:

n_i = número total de respuestas para el atributo i de las herramientas.

Se calcula el índice de satisfacción general para el atributo i de la herramienta:

$$ICS_i = (C_i * 25) - 25$$

Esta formula nos permite obtener un índice que toma valores entre 0 y 100, por lo que es posible también expresarlo entre 0 y 1, donde 0 es totalmente insatisfecho y 1 es totalmente satisfecho.

2.8. Software utilizado

SPSS 11.0 for Windows es un conjunto de programas orientados a la realización de análisis estadísticos aplicados a las ciencias sociales. Nos permite realizar análisis y gráficos estadísticos sin tener que conocer la mecánica de los cálculos ni la sintaxis de los comandos del sistema. Comparado con otros programas, es más intuitivo y fácil de aprender. Su desventaja es que es menos flexible y con menos procedimientos avanzados que otros programas comerciales.

SPSS es bueno a la hora de organizar y analizar datos. Se puede ordenar datos, calcular nuevos datos y realizar una gran variedad de análisis estadísticos. En teoría el tamaño de los ficheros de datos que SPSS puede manejar no está limitado por lo que puede trabajar con ficheros

grandes. Esta versión también permite el manejo cómodo de ficheros, la personalización de los informes, y el cortar y pegar en otros programas.

Hay dos tipos de archivos asociados a SPSS:

- Archivos de datos: tienen extensión .sav y están en formato SPSS.
- Archivos de texto: tienen extensión .sps (archivos de sintaxis) o .spo (archivos de resultados).

2.8.1. SPSS breve descripción

Los pasos básicos en el análisis de datos consisten en:

- Introducir los datos, manualmente o recurriendo a un archivo ya existente.
- Seleccionar un procedimiento estadístico.
- Seleccionar las variables para el análisis, las variables que podemos usar en cada procedimiento se muestran en un cuadro de diálogo del que se seleccionan.
- Ejecutar el procedimiento y ver los resultados, los resultados aparecen en una ventana de resultados y se pueden guardar como archivos con extensión .spo. Los gráficos se pueden modificar en la ventana del editor de gráficos que se presenta cuando se da doble clic sobre dicho gráfico.

2.8.2. Vistas en SPSS

La primera presentación es de una tabla de datos, donde se deberán introducir los datos de cada problema o leerlos de un fichero. Corresponde al Editor de datos.

Con el editor de datos podemos crear nuevos archivos o modificar los existentes. No se puede tener más de un archivo de datos abierto al mismo tiempo en la misma sesión de SPSS. Dentro del editor de datos, dos vistas son posibles:

- **Vista de datos,** muestra los valores de datos reales o las etiquetas de valor definidas:

  Las filas son casos. Cada fila representa un caso u observación.

  Las columnas son variables. Cada columna representa una variable o característica que se mide.

  Las casillas contienen valores numéricos o de cadena, siendo éste un valor único de una variable para cada caso. A diferencia de una hoja de cálculo, las casillas del editor de datos no pueden contener fórmulas.

- **Vista de variables**, contiene descripciones de los atributos de cada variable del archivo de datos. Aquí:

  Las filas son variables.

Las columnas son atributos o características de las variables.

Cambiamos de una vista a otra a través de las pestañas en la parte inferior de la ventana.

Otras ventanas irán apareciendo a medida que vayamos realizando nuestro análisis, podrán contener gráficos (Editor de gráficos), informes con los resultados, etc.