UNIVERSIDAD AUTONOMA DE NUEVO LEON

FACULTAD DE INGENIERIA MECANICA Y ELECTRICA
DIVISION ESTUDIOS DE POST-GRADO

EL IMPACTO DE LA ENSEÑANZA DE METRICAS DE SOFTWARE EN LA IMPLEMENTACION DE UN SISTEMA COMPUTACIONAL

T E S I S

EN OPCION AL GRADO DE MAESTRO EN CIENCIAS DE LA ADMINISTRACION CON ESPECIALIDAD EN SISTEMAS

POR

ING. EDGAR DANILO DOMINGUEZ VERA

SAN NICOLAS DE LOS GARZA, N. L., ENERO DE 1999

1999

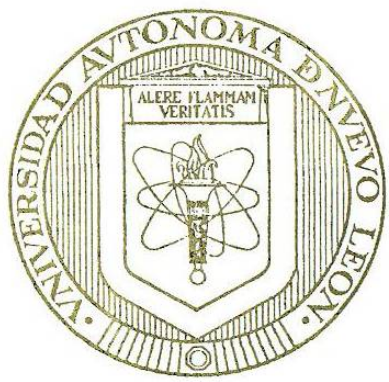
EL IMPACTO DE LA ENSEÑANZA DE METRICAS DE SOFTWARE EN LA IMPLEMENTACION DE UN SISTEMA COMPUTACIONAL

E.D.D.V.

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

DIVISIÓN DE ESTUDIOS DE POST-GRADO



EL IMPACTO DE LA ENSEÑANZA DE MÉTRICAS DE SOFTWARE EN LA IMPLEMENTACIÓN DE UN SISTEMA COMPUTACIONAL

TESIS
EN OPCION AL GRADO DE MAESTRO EN CIENCIAS DE
LA ADMINISTRACIÓN CON ESPECIALIDAD
EN SISTEMAS

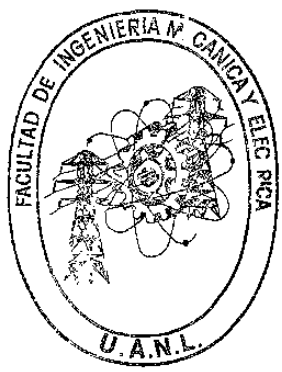
POR
ING. EDGAR DANILO DOMÍNGUEZ VERA

SAN NICOLAS DE LOS GARZA, N.L. ENERO DE 1999

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

DIVISIÓN DE ESTUDIOS DE POST-GRADO



EL IMPACTO DE LA ENSEÑANZA DE MÉTRICAS DE SOFTWARE EN LA IMPLEMENTACION DE UN SISTEMA COMPUTACIONAL

TESIS
EN OPCIÓN AL GRADO DE MAESTRO EN CIENCIAS DE
LA ADMINISTRACIÓN CON ESPECIALIDAD
EN SISTEMAS

POR
ING. EDUARDO DOMÍNGUEZ VERA

SAN NICOLÁS DE LA GARZA, N.L. ENERO DE 1999

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN
FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA
DIVISIÓN DE ESTUDIOS DE POST-GRADO

Los miembros del comité de tesis recomendamos que la tesis de "EL IMPACTO DE LA ENSEÑANZA DE MÉTRICAS DE SOFTWARE EN LA IMPLEMENTACIÓN DE UN SISTEMA COMPUTACIONAL" realizada por el Ing. Edgar Danilo Domínguez Vera sea aceptada para su defensa como opción al grado de Maestro en Ciencias de la Administración con especialidad en Sistemas.

El comité de Tesis

Asesor
Dr. José Luis Martínez Flores

Coasesora
Dra. Ada Margarita Álvarez Socarrás

Coasesora
M.C. Rosa María Reséndez Hinojosa

Vo.Bo.
M.C. Roberto Villarreal Garza
División de Estudios de Post-grado

San Nicolas de los Garza, N.L. diciembre de 1998

DEDICATORIAS

A mis padres:

Sr. Belisario Domínguez Villanueva (†) y Dulce María Vera Ibañez.

A mi abuelita:

Sra. María Isabel Ibañez Bache, por ti se que hay ternura en el mundo.

A mis hermanos:

José Manuel Belisario, Julio César, Javier, Jorge Ernesto, Valentín Belisario, Dulce Nelly y Antonio.

A mi tíos:

Ruby Laura Vera Ibañez y Francisco López Arostegui, por su amistad.

A mis compañeros y amigos:

En especial a los de la E.S.B.O. # 8, y a Luis Hernández, el orgullo nuestro, por ser el mejor futbolista de México.

A mi futura esposa e hijos:

Por que no creo que detrás de un hombre hay una gran mujer, para mí, la gran mujer está siempre al lado de su esposo, ¡nunca atrás!.

A mis Maestros y Amigos de la Facultad de Filosofía y Letras

A quiénes comparten el amor, la fe y la esperanza del Sr. Jesucristo

AGRADECIMIENTOS

A Dios, por su amor y misericordia.

No agradecer al Ing. José Antonio González Treviño, sería tanto como aceptar que mi vocabulario personal no empieza con esta noble palabra: Agradecimiento. Por su apoyo y amistad. GRACIAS…

Al Ing. Cástulo Vela Villarreal, por su confianza y apoyo.

A los Ing. Alfonso Molina Rodríguez y David Garza Garza por su amistad.

Lic. Rogelio Moreno Obregón, Ing.Arturo Borjas Roacho, Ing. Pablo Rodríguez Tristán. Por su amistad y apoyo en los inicios de mi carrera profesional, y sobretodo, cuando más lo necesité.

M.C. Roberto Elizondo Villarreal y M.C. Jesús José Meléndez Olivas, por el idealismo que compartimos.

A José Antonio Moreno Barrios y Arturo del Angel Ramírez por su amistad

Al Dr. José Luis Martínez Flores, por sus consejos y ayuda en la realización de esta tesis.

... Al hombre del siglo XX le ha sido muy difícil no quedar a la zaga. En tanto nuevos medios de comunicación y transporte han hecho cada vez más chico el planeta, la vasta expansión de los conocimientos le ha hecho imposible tener ***una visión global del mundo****. La culminación de la Revolución Industrial, el progreso de la tecnología electrónica y la necesidad de especialización han* ***fragmentado aún más su visión****...*

William Fleming,1996. Arte, Música e Ideas. p. 333

PRÓLOGO

Hoy en día, la competitividad del mercado ha llevado a las empresas a buscar mecanismos que les permitan medir la calidad de los productos y servicios que ofrecen; la medición de la calidad de un producto o servicio está vinculada a la productividad de los trabajadores que intervienen en la producción o desarrollo de dicho bien.

Hoy en día, las herramientas para medir la productividad de un grupo de programadores y para medir la calidad del software, son escasas. Dada esta escasez, se pretende contribuir para encontrar una herramienta que permita llenar el vacío que ahora existe.

Por otro lado, es común que algunos analistas de sistemas, al menos en nuestro entorno regional, hagan estimaciones de software en base a su experiencia. Esto puede dar buenos resultados si el analista tiene mucha experiencia, pero, en primer lugar, deja ver que falta una metodología formal que permita hacer tales estimaciones de software, y en segundo lugar, no es posible medir el grado de confianza que unas estimaciones puedan tener, cuando se hacen sólo en base a la experiencia.

Una de la fases más descuidada del desarrollo de software es la relativa al mantenimiento. Así, podemos encontrar que a algunos analistas sólo les interesa que el software "funcione", y se olvidan que el desarrollo de software está ligado al establecimiento de una relación con el cliente a postventa.

Por otro lado, una buena parte de los programadores creen que es suficiente hacer programas que arrojen los resultados correctos y se olvidan que un buen programa es el que es fácil de entender, fácil de modificar y desde luego que arroje resultados correctos.

Todo programador tiene la responsabilidad social de preguntarse a sí mismo, si el programa que ha desarrollado, puede ser entendido por una persona que tenga conocimiento medios-avanzados del lenguaje en el que se ha desarrollado. Esto con la finalidad de buscar que los programas se hagan de tal forma que permitan la facilidad de mantenimiento.

Otro vacío que se ha encontrado, es la falta de herramientas que permitan el establecimiento de estándares tal que una organización pueda pedir a los desarrolladores de software se ajusten a ellos.

La literatura previa nos indica que se ha demostrado que los estudiantes que han recibido una enseñanza en las métricas de software producen programas que exhiben menos complejidad, requieren menos tiempo de codificación y prueba, y además es más fácil de darle mantenimiento. Esto no se ha comprobado en nuestro entorno regional y menos teniendo como herramienta de desarrollo el lenguaje FOXPRO.

En 1997, el lenguaje más solicitado por las empresas regiomontanas, en las ofertas de empleo, fue precisamente el FOXPRO, es por eso, que en la presente tesis se decidió investigar cuál es el impacto que tiene la enseñanza de las métricas de software sobre la calidad del software, teniendo como herramienta de desarrollo el lenguaje anteriormente mencionado.

El impacto encontrado fue contundente, por lo que podemos decir, he aquí una herramienta que ayuda a mejorar la calidad del software, que permite medir la productividad de los programadores y establecer estándares de calidad.

Es importante recalcar que el instrumento de medición utilizado, fue desarrollado en una investigación anterior y aunque se le hizo una muy mínima actualización, podemos decir que tal instrumento está completamente validado, ese instrumento es llamado: Analizador de Código.

Por último, espero que esta tesis pueda dar algunas guías para hacer mejores estimaciones en cuanto a la medición de la productividad de los programadores y la calidad de software.

Edgar Danilo Domínguez Vera

Septiembre de 1998.

ÍNDICE GENERAL

CONTENIDO PÁGINA

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

SÍNTESIS

En la presente tesis se determina que impartir un curso de métricas de software a los estudiantes de Sistemas Computacionales, sí contribuye para que se genere código fuente de mejor calidad.

La metodología de la investigación se realizó de la siguiente manera:

1. La herramienta de desarrollo de software que se utilizó fue el lenguaje FOXPRO2.6 para Windows.
2. Se hicieron un grupo experimental y uno de control constituidos por seis estudiantes cada uno. Los estudiantes son alumnos de la carrera de I.A.S. en la FIME-UANL, del quinto semestre.
3. Cada estudiante desarrollo10 programas en FOXPRO2.6 para Windows; los programas fueron los mismos para ambos grupos.
4. Se utilizó un analizador de código fuente realizado en una investigación anterior.
5. Se hicieron pruebas Z para comparar medias entre los grupos. La comparación se hizo en cuanto el volumen de los programas, el nivel de los programas, el contenido de inteligencia en los programas, el esfuerzo de programación, el tiempo estimado de programación y el nivel del lenguaje.

Mediante un análisis de los datos obtenidos se lograron resultados que permiten la aceptación de 4 de las 6 hipótesis propuestas con diferentes grados de confiabilidad. Es importante destacar que la dos hipótesis no aceptadas, si bien no hay suficiente prueba estadística que permita su aceptación sí hay tendencias a favor de cada una de ellas.

En conclusión, los resultados encontrados fueron:

1. Se encontró que en la Hipótesis que establece que la media del volumen del programa de los programas hechos por el grupo de experimental es menor que

la media del volumen del programa de los programas hechos por el grupo de control, hay una tendencia a favor del 61.83%, esto con un nivel de confianza del 80%. Sin embargo, no hay suficiente prueba estadística que permita aceptar dicha hipótesis.

2. Hay suficiente prueba estadística, con una confiabilidad del 90%, que permite aceptar que la media del nivel del programa de los programas hechos por el grupo experimental es mayor a la media del nivel del programa de los programas hechos por el grupo de control.
3. Se encontró que en la hipótesis que establece que la media del contenido de inteligencia de los programas hechos por el grupo experimental es mayor a la media del contenido de inteligencia de los programas hechos por el grupo de control, hay una tendencia a favor del 34.48%, esto con una confiabilidad del 80%. Sin embargo, no hay suficiente prueba estadística que permita aceptar dicha hipótesis.
4. Hay suficiente prueba estadística, con una confiabilidad del 80%, que permite aceptar que la media del esfuerzo de programación de los programas hechos por el grupo experimental es menor a la media del esfuerzo de programación de los programas hechos por el grupo de control.
5. Hay suficiente prueba estadística, con una confiabilidad del 80%, que permite aceptar que la media del tiempo estimado de programación de los programas hechos por el grupo experimental es menor a la media del tiempo estimado de programación de los programas hechos por el grupo de control.
6. Hay suficiente prueba estadística, con una confiabilidad del 90%, que permite aceptar que la media del nivel del lenguaje de los programas hechos por el grupo experimental es mayor a la media del nivel del lenguaje de los programas hechos por el grupo de control.

CAPÍTULO 1

INTRODUCCIÓN

1.0 PLANTEAMIENTO DEL PROBLEMA

Una de las áreas más descuidadas, en el desarrollo de software, es el mantenimiento del mismo. El mantenimiento del software existente puede llevarse hasta el 70% del esfuerzo gastado por una organización de desarrollo, por eso es necesario que se desarrolle con calidad.

La calidad del software se entiende como la facilidad con la que se le puede dar mantenimiento, entendiendo esto último como la facilidad de comprender, corregir y/o mejorar el software [Pressman,1993].

Hoy en día, la globalización mundial ha llevado a la administración moderna a medir la calidad y la productividad dentro de sus compañías para poder ser competitivos. En particular, las compañías que desarrollan software han realizado esfuerzos dirigidos a medir la calidad del mismo, así como para medir la productividad del personal involucrado en tales proyectos.

En la mayoría de los desafíos técnicos que implique la producción de un bien o servicio, la medición y las métricas ayudan a entender tanto el proceso técnico que se utiliza para desarrollar un producto, como el propio producto. El proceso se mide para intentar mejorarlo. El producto se mide para intentar aumentar su calidad.

Sin embargo, la medición del software es más controvertida por ser un elemento lógico y no físico, por lo que es más difícil y hasta subjetivo determinar: i) las métricas apropiadas para el proceso y para el producto, ii) la utilización de los datos que se recopilan y iii) las medidas para comparar gente, procesos o producto.

Por otra parte, debido a la naturaleza subjetiva de la calidad del software es común obtener medidas poco precisas, por lo que debemos estar conscientes que no se puede esperar medir la calidad del software de forma exacta, ya que cada medida es parcialmente imperfecta.

Sin embargo, para poder medir directamente la calidad del software, existe un conjunto de métricas que se pueden aplicar para garantizar cuantitativamente su calidad. En todos sus casos, las métricas representan medidas indirectas, es decir, nunca medimos realmente la calidad, sino algunas de sus manifestaciones [Pressman,1993].

Así mismo, para medir la calidad del software se deben tomar en cuenta los factores que lo afectan en su desarrollo, estos incluyen: Su tipo, su tamaño, el lenguaje de implementación, la experiencia de los programadores involucrados, las técnicas de programación involucradas y el ambiente computacional [Martínez,1994].

Dentro de las métricas propuestas para medir la calidad del software está la Teoría de Halstead sobre la Ciencia del Software, que es probablemente la más conocida y la más estudiada de las métricas que se pueden aplicar para garantizar cuantitativamente la calidad del software [Pressman,1993].

La Ciencia del Software [Halstead,1977] es un modelo del proceso de programación que se basa en un número manipulable de los principales factores que afectan la programación. Esto ofrece una guía hacia estimadores que pueden ser útiles a los administradores de proyectos de software. Los resultados de esta teoría se han utilizado y se siguen utilizando [Wrigley,1991], [Bowman,1990] con sus

correspondientes críticas y evaluaciones [Shen,1983], [Weyucker,1988] y [Ramamurthy,1988].

Los resultados de la Ciencia del Software de Halstead se basan en muestras de programas escritos en lenguaje máquina, ensamblador y de tercera generación; la cual usa un conjunto de primitivas de medida que se pueden obtener una vez que se ha generado el código o estimar una vez que se ha terminado el diseño. Halstead [1977] usa las primitivas de medidas para desarrollar expresiones para: **la longitud total de un programa, el mínimo volumen potencial de un algoritmo, el volumen real, el nivel del programa, el nivel del lenguaje y otras características tales como el esfuerzo de desarrollo, el tiempo de desarrollo e incluso el número previsto de fallos en el software**.

Se ha demostrado empíricamente que el estimador de la longitud de un programa es válido para lenguajes de cuarta generación, tales como: DBASE y FOXPRO2 [Martínez,1994]. La investigación que determinó lo anterior consistió en realizar un analizador de código que examinó un grupo de programas en código fuente. La validez de los resultados obtenidos es, en el caso de FOXPRO2, para programas que están en el rango de 12 a 4,610 como longitud de un programa [Martínez,1994].

Por lo tanto, disponemos de una herramienta que nos permite medir la calidad del software desarrollado en FOXPRO2.

Por otra parte, desde un punto de vista académico, nos preguntamos ¿qué debemos enseñar a nuestros alumnos para desarrollar software de calidad?.

En EE.UU. se hizo un estudio tratando de responder esta pregunta y se llegó a la conclusión de que los estudiantes que han recibido enseñanza en métricas de software, producen (desarrollan) programas que exhiben menos complejidad, requieren menos tiempo de codificación y prueba, y además, es más fácil de darle mantenimiento [Bowman,1990].

Por lo anterior, es importante que se haga un estudio similar para saber si la enseñanza de métricas de software produce el mismo efecto en nuestro entorno, es decir, deseamos saber si la enseñanza de métricas de software contribuye para que un grupo de

estudiantes genere código fuente con calidad. Con esto, daremos pasos firmes hacia la medición formal de la calidad del software.

1.1 JUSTIFICACIÓN DEL ESTUDIO

La importancia de la presente investigación es que actualmente, en la carrera de Ingeniero Administrador de Sistemas de la F.I.M.E. - U.A.N.L. los temas correspondientes a la Ingeniería del Software y las Métricas de Software no han tenido el lugar e importancia que requieren. Este estudio se realiza con el fin de justificar, empíricamente, que tales temas deben ser incluidos en el programa de estudios de tal carrera, ya que una de las áreas de oportunidad del I.A.S es precisamente el desarrollo de software.

En el área educativa, se ayudará a determinar cuáles son los temas que se deben incluir en un curso que vaya dirigido a estudiantes de sistemas de computación, al personal de desarrollo de software y a todo profesionista que esté interesado en la generación de software de calidad.

En la parte financiera, se contribuirá para que se encuentren áreas de oportunidad que permitan bajar los costos de desarrollo y mantenimiento de software.

En el área administrativa, se contribuirá para que la medición de la productividad del personal que se desenvuelve en un proyecto de desarrollo de software se realice con bases objetivas y además para que se fortalezcan las bases que permitan la planeación de proyectos de desarrollo de software en cuanto a presupuestos de costo y tiempo.

1.2 OBJETIVO DEL ESTUDIO

Dado lo anterior, el Objetivo General de la presente tesis es determinar si impartir un curso de métricas de software a los estudiantes de Sistemas Computacionales contribuye para que se genere código fuente de mejor calidad.

La presente tesis trata de dar respuesta a la pregunta anteriormente formulada mediante el logro de los siguientes objetivos específicos: Determinar si impartir un curso de Métricas de Software a los estudiantes de Sistemas Computacionales contribuye para:

Objetivo 1: Que se genere código cuya media del volumen del programa sería más bajo que si no se hubiera impartido el curso.

Objetivo 2: Que se genere código cuya media de nivel del programa sería más alta que si no se hubiera impartido el curso.

Objetivo 4: Que se genere código cuya media del contenido de inteligencia sería más alta que si no se hubiera impartido el curso.

Objetivo 5: Que se genere código cuya media del esfuerzo de programación sería más bajo que si no se hubiera impartido el curso.

Objetivo 6: Que se genere código cuya media del tiempo estimado de programación sería más bajo que si no se hubiera impartido el curso.

Objetivo 7: Que se genere código cuya media del nivel del lenguaje sería más alta que si no se hubiera impartido el curso.

1.3 LIMITACIONES DEL ESTUDIO

Se puede clasificar a los programas por su longitud (*N*) como pequeños (***N*<=100**), medianos (***N*>100** y ***N*<=500**) , grandes (***N*>500** y ***N*<=1500**) y muy grandes (***N*>1500**), entendiéndose como longitud de un programa (*N*) a la suma de los operadores[1] y operandos[2] que se utilizan en la codificación de un algoritmo.

Por lo tanto, una de las limitaciones de este estudio es que la dispersión de los programas, en cuanto a la longitud (*N*), está restringida a programas que son medianos.

De este modo para el grupo experimental el 3.33% de los programas son pequeños, el 88.33% son medianos y el 8.33% son grandes. Para el grupo de control el 1.66% de los programas son pequeños, el 88.33% son medianos y el 10% son grandes

[1] Operador: Cualquier símbolo en un programa que especifique una acción algorítmica.
[2] Operando: Cualquier símbolo en un programa que represente datos.

Se cree que si los programas tienen una mejor dispersión, en cuanto a N, y si tienen rangos más grandes se puede hacer un mejor análisis [Martínez,1994]

Otra limitación es que la función, de todos los programas, es para hacer reportes, es decir, lista que siempre presenta los datos con el mismo formato [Navarro,1997]. Entonces, en esta investigación no se incluyen programas de actualizaciones, consultas, administración de bases de datos, funciones matemáticas, menús y otros. Se comenta lo anterior porque se ha comprobado que la función de un programa influye en la correlación entre N y $N^{\wedge}$[3], y en el nivel del lenguaje[4] [Baez-Casselyn,1997].

También, podemos agregar dentro de la limitaciones, que si bien la muestra en cuanto a número de programas se considera, estadísticamente hablando, grande; la muestra, en cuanto a número de estudiantes en cada grupo (el experimental y el de control), es pequeña (seis estudiantes por cada grupo).

1.4 RESUMEN DEL CAPÍTULO

En este capítulo se planteó el problema en el que está inmerso el estudio, la justificación, el objetivo general, los objetivos específicos y las limitaciones de la investigación. En el siguiente capítulo se da a conocer el marco teórico.

[3] Longitud Estimada de un Programa [Pressman,1993]

[4] Expresa el poder del lenguaje [Martínez,1994]. Los lenguaje con alto nivel permiten una especificación del código a un mayor nivel de abstracción que el lenguaje ensamblador (orientado a la máquina) [Pressman,1993]

CAPÍTULO 2

ANTECEDENTES

2.0 INTRODUCCIÓN

Dada la importancia que las métricas de Halstead tienen, se explicará el lugar en que se ubican dichas métricas, así como los conceptos que nos conducen a su significado.

El presente capítulo se conducirá de la siguiente forma. En 2.1 se dará una explicación del concepto de software, de su desarrollo y de sus problemas. En 2.2 se dará una explicación de lo que es la ingeniería de software. En 2.3 se dará una explicación de lo que son las métricas de software, su clasificación y sus dimensiones. En 2.4 se dará una explicación de las métricas de calidad del software, y su clasificación. y en 2.5 se dará un resumen del capítulo.

2.1 SOFTWARE

2.1.1 CONCEPTO DE SOFTWARE

El Software es un concepto muy difícil de definir, aunque se le diera una definición formal se podrían encontrar otras definiciones más completas. Se necesita algo más que una definición formal para poder comprender lo que es el software. Sin embargo, podemos definir software como: Programas o conjunto de instrucciones que dirigen las operaciones de procesamiento de información ejecutadas por medio del hardware [Athey,1988] .

Para comprender lo que es el software, es importante examinar sus características que lo hacen ser diferente de otras cosas que el hombre puede construir, como el hardware. El software es un elemento del sistema que es lógico , en lugar de físico, por lo tanto:

1) El software se desarrolla, no se fabrica en un sentido clásico.
2) El software no se estropea con el tiempo, en cambio el hardware sí se estropea (ver Fig. 2.1 y Fig. 2.2) [Pressman,1993]

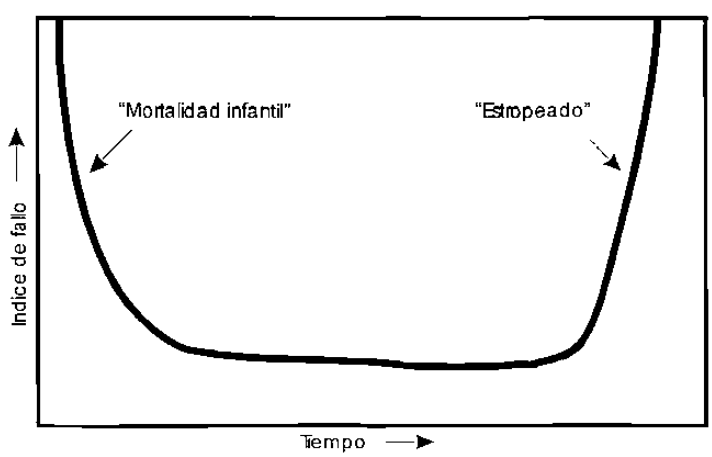


Fig. 2.1 Curva de Fallas del Hardware.

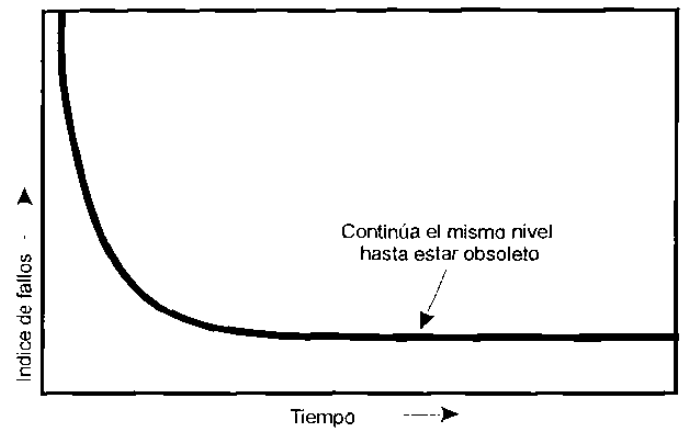


Fig. 2.2 Curva Idealizada de fallas del Software.

Sin embargo, durante su vida, el software sufre cambios (mantenimiento). Conforme se hacen los cambios, es bastante probable que se introduzcan nuevos defectos, haciendo que la curva fallos tenga picos como se ve en la figura 2.3.

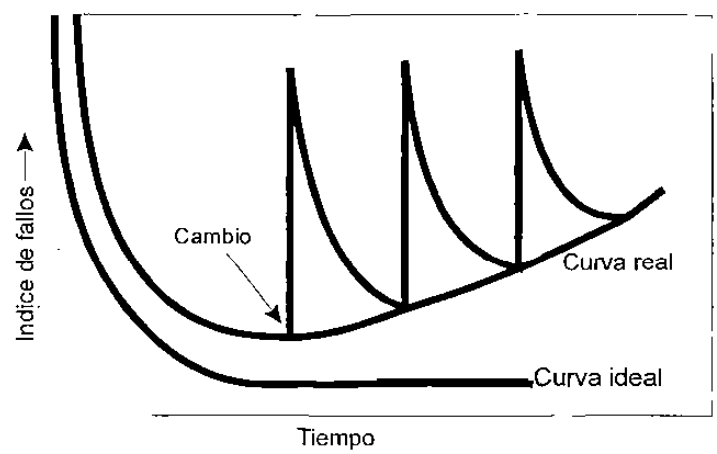


Fig. 2.3 Curva Real de Fallas del Software

3) La mayoría del software se construye a la medida, en vez de ensamblar componentes existentes. Esta característica tiende a desaparecer. La reusabilidad del software va a ser en la presente década de los noventa, uno de los principales contribuidores en la productividad y calidad del software [Yourdon,1992].

2.1.2 DESARROLLO DE SOFTWARE

El software se desarrolla mediante un lenguaje de programación que tiene un vocabulario limitado, una gramática definida explícitamente y reglas bien formadas de sintaxis y semántica. Las clases de lenguajes que se utilizan actualmente son los lenguajes de alto nivel de tercera y cuarta generación.

2.1.3 PROBLEMAS CON EL DESARROLLO DE SOFTWARE

Los problemas que afligen al desarrollo de software se pueden caracterizar bajo perspectivas diferentes, pero se centran en las siguientes [Pressman,1993] .

1) La planificación y estimación de costos son frecuentemente imprecisas.
2) La productividad de los desarrolladores de software no satisface lo que los usuarios demandan.
3) La calidad del software es cuestionable.

2.1.4 CAUSAS DE LOS PROBLEMAS DEL DESARROLLO DE SOFTWARE

Los problemas con el desarrollo de software se han producido por la propia naturaleza del software y por los errores de las personas encargadas de su desarrollo. Como se dijo anteriormente, el software es un elemento lógico en vez de físico y por lo tanto, el éxito se mide por la calidad de él como un todo.

El software no se rompe. Si se encuentran fallas, lo más probable es que se introdujeran inadvertidamente durante el desarrollo y no se detectaran durante la prueba. Reemplazamos las partes defectuosas durante el mantenimiento del software, es decir, el mantenimiento incluye normalmente la corrección o modificación del diseño.

Frecuentemente, los responsables del desarrollo de software han sido ejecutivos de nivel medio y alto, sin conocimientos de software. El administrador de software debe comunicarse con todas las partes implicadas en el desarrollo del software: clientes, desarrolladores, equipo de soporte, y otros. La comunicación puede romperse debido a

que se comprenden mal las características especiales del software y los problemas particulares asociados con su desarrollo.

Los programadores (ingenieros de software) han tenido muy poco entrenamiento formal en las nuevas técnicas del desarrollo de software. Cada individuo enfoca su tarea de programación con la experiencia obtenida en trabajos anteriores. Algunas personas desarrollan un método ordenado y eficiente del desarrollo de software mediante prueba y error, pero muchos desarrollan malos hábitos que dan como resultado una pobre calidad y mantenibilidad del software[1].

2.2 LA INGENIERÍA DEL SOFTWARE

2.2.1 DEFINICIÓN

No existe un enfoque que sea el mejor para la solución de los problemas del software. Sin embargo, mediante la combinación de métodos completos para todas las fases del desarrollo de software, mejores herramientas para automatizar estos métodos, bloques de construcción más potentes para la implementación de software, mejores técnicas para la garantía de calidad del software y una filosofía predominante para la coordinación, control y administración, podemos encontrar una disciplina para el desarrollo de software: Ingeniería de Software.

Una definición de ingeniería de software es la siguiente:

La ingeniería de software es el enfoque sistemático hacia la especificación, desarrollo, operación, mantenimiento y retiro del software [Sigwart,1990]

Como se comentó en el concepto de software, se necesita comprender lo que es la ingeniería de software más que su definición formal. La ingeniería de software abarca un conjunto de tres elementos clave: métodos, herramientas y procedimientos. Estos

[1] Mantenibilidad: la facilidad con que se le puede dar mantenimiento al software

elementos ayudan al administrador a controlar el proceso de desarrollo de software y a suministrar, a los que practican dicha ingeniería, las bases para construir software de alta calidad de una forma productiva [Pressman,1993].

Los **métodos** de la ingeniería de software indican "cómo" construir técnicamente el software. Los métodos abarcan un amplio espectro de tareas que incluyen: planificación y estimación de proyectos, análisis de los requisitos del sistema y del software, diseño de estructura de datos, arquitectura de programas y procedimientos algorítmicos, codificación, prueba y mantenimiento.

Las **herramientas** de la ingeniería de software suministran un soporte automático o semiautomático para los métodos. Cuando se integran las herramientas de tal manera que la información creada por una herramienta pueda ser usada por otra, se establece un sistema para el soporte del desarrollo de software llamado ingeniería de software asistida por computadora (C.A.S.E.)

Los **procedimientos** de la ingeniería de software son el aglutinante de los métodos y herramientas, que facilita un desarrollo racional y oportuno del software. Los procedimientos definen la secuencia en la que se aplican los métodos, las entregas (documentos, reporte, formas, etc.) que se requieren, los controles que ayudan a asegurar la calidad y coordinar los cambios, y las directrices que ayudan a los administradores de software a evaluar el progreso.

2.3 MÉTRICAS DE SOFTWARE

La medición es muy común en el mundo de la ingeniería. Medimos potencias de consumo, pesos, dimensiones físicas, temperaturas, voltajes, etc. Desafortunadamente, la medición se aleja de lo común en el mundo de la ingeniería de software. Encontramos dificultades en ponernos de acuerdo sobre qué medir y cómo evaluar las medidas.

Algunas de las razones para medir el software son las siguientes:

1) Para indicar la calidad del producto.

2) Para evaluar la productividad de la gente que desarrolla el producto.

3) Para evaluar los beneficios (en términos de productividad y calidad) que se derivan del uso de nuevos métodos y herramientas de la ingeniería de software.
4) Para establecer una línea de base para la estimación.
5) Para ayudar a justificar el uso de nuevas herramientas o la formación de otras herramientas.

Las medidas de software pueden clasificarse en dos categorías: medidas directas y medidas indirectas. Entre las medidas directas se encuentran: el costo y el esfuerzo requerido para construir el software, el número de líneas de código, etc. Entre las medidas indirectas del proceso de ingeniería del software se encuentran: la calidad y funcionalidad del software, la eficiencia y facilidad de mantenimiento.

El campo de las métricas del software se puede clasificar en dos dimensiones (Fig. 2.4) [Pressman,1993]. En una dimensión tenemos las siguientes métricas:

1) Métricas de productividad. Estas métricas se centran en el rendimiento del proceso de la ingeniería de software.
2) Métricas de calidad. Estas métricas proporcionan una indicación de cómo se ajusta el software a los requisitos implícitos y explícitos del cliente.
3) Métricas técnicas. Estas métricas se centran en las características del software más que en el proceso a través del cual ha sido desarrollado el software.

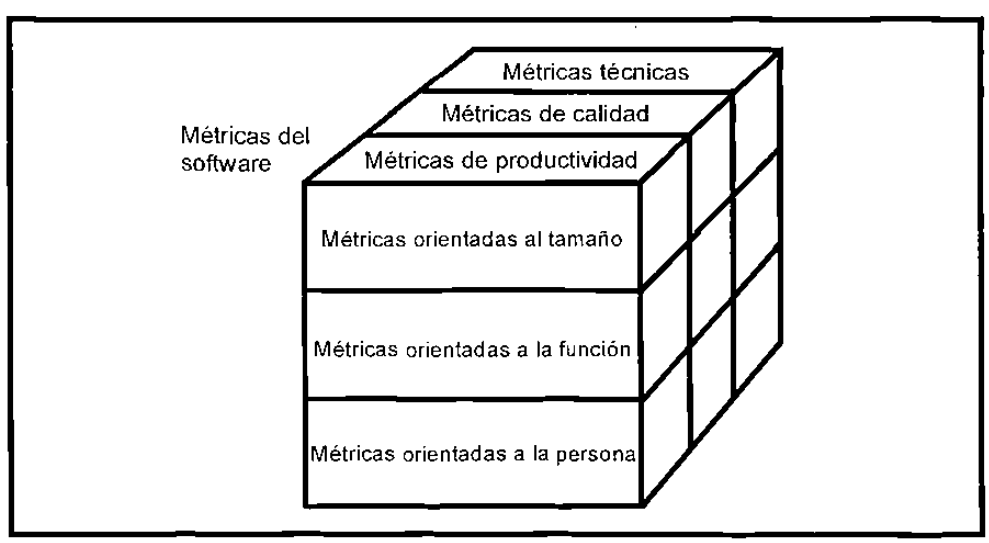


Fig. 2.4 Clasificación de Métricas.

En otra dimensión tenemos las siguientes métricas:

1) Métricas orientadas al tamaño. Estas métricas son medidas directas del software y del proceso por el cual se desarrolla. Por ejemplo líneas de código (LOC), esfuerzo y costo.
2) Métricas orientadas a la función. Estas métricas son medidas indirectas del software y del proceso por el cual se desarrolla. En lugar de calcular las LOC, las métricas orientadas a la función se centran en la funcionalidad o utilidad del programa.
3) Métricas orientadas a las personas. Esta métricas proporcionan información sobre la forma en que la gente desarrolla el software y sobre el punto de vista humano de la efectividad de las herramientas y métodos.

2.4 MÉTRICAS DE CALIDAD DEL SOFTWARE

2.4.1 CALIDAD DEL SOFTWARE

Pressman [1993] define la calidad del software como la concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente.

La definición de calidad del software hace énfasis en tres puntos importantes:

1) Los requisitos (funcionales) del software son la base de las medidas de la calidad. La falta de concordancia con los requisitos es una falta de calidad.
2) Los estándares especificados definen un conjunto de criterios de desarrollo que guían la forma en que se aplica la ingeniería de software. Si no se siguen esos criterios, casi siempre habrá falta de calidad.
3) Existe un conjunto de requisitos implícitos que a menudo no se mencionan (por ejemplo, el deseo de un buen mantenimiento). Si el software se ajusta a sus requisitos explícitos pero falla en alcanzar los requisitos implícitos, la calidad del software deja mucho que desear.

2.4.2 FACTORES DE CALIDAD DEL SOFTWARE

Los factores que afectan la calidad del software se clasifican, principalmente, en dos categorías:

1) Factores que pueden ser medidos directamente, tal como los errores/LOC/unidad tiempo.
2) Factores que sólo pueden ser medidos indirectamente, tal como la facilidad de uso o de mantenimiento.

Una clasificación de los factores que afectan la calidad del software ha sido propuesta por McCall [1977]. Estos factores de calidad del software, que se pueden ver en la Fig. 2.5, se centran en tres aspectos importantes de un producto de software: sus características operativas, su capacidad de soportar los cambios y su adaptabilidad de nuevos entornos.

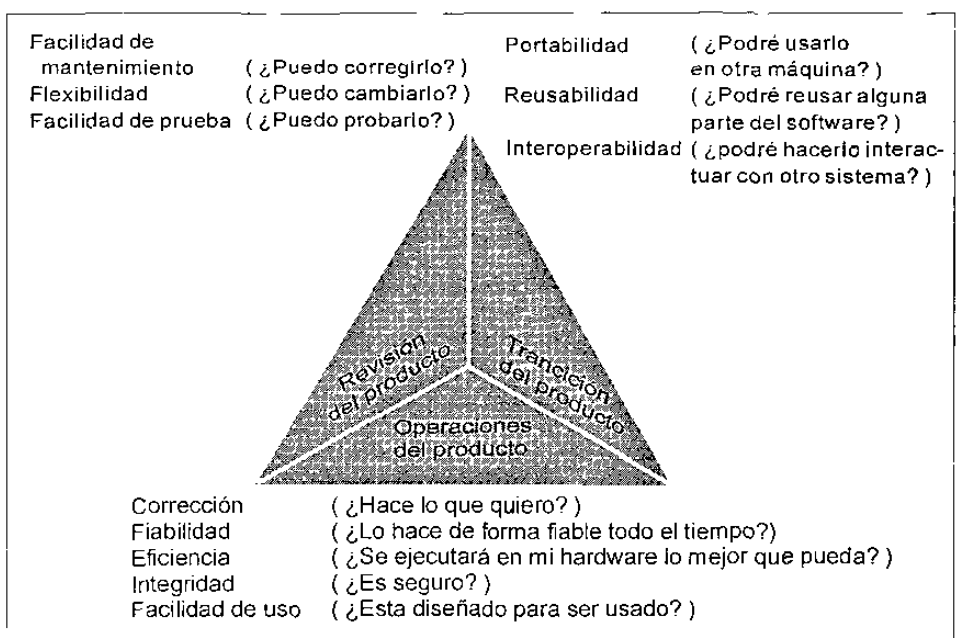


Fig. 2.5 Factores de Calidad del Software de McCall

Es difícil, y en algunos casos imposible, desarrollar medidas directas de los anteriores factores de calidad. Por tanto, se define un conjunto de métricas usadas para medir en forma indirecta las factores de calidad del software.

Existen dos tipos de métricas:

1) Métricas Cualitativas. Esta métricas sólo pueden ser medidas en forma subjetiva.
2) Métricas Cuantitativas. Estas métricas sí pueden medirse en forma objetiva.

2.4.3 MÉTRICAS CUALITATIVAS DE LA CALIDAD DEL SOFTWARE

Entre las métricas cualitativas de la calidad del software encontramos las siguientes métricas definidas por McCall [1977]:

Facilidad de Auditoría. La facilidad con que se puede comprobar la conformidad con los estándares.

Exactitud. La precisión de los cálculos y del control.

Normalización de las comunicaciones. El grado en que se usan el ancho de banda, los protocolos y las interfases estándar.

Completitud. El grado en que se ha conseguido la total implementación de las funciones requeridas.

Concisión. Lo compacto que es el programa en términos de líneas de código.

Consistencia. El uso de un diseño uniforme y de técnicas de documentación a lo largo del proyecto de desarrollo de software.

Estandarización de los datos. El uso de estructuras de datos y de tipos estándar a lo largo de todo el programa.

Tolerancia de errores. El daño que se produce cuando el programa encuentra un error

Eficiencia en la ejecución. El rendimiento en tiempo de ejecución de un programa.

Facilidad de Expansión. El grado en que se puede ampliar el diseño arquitectónico, de datos o procedimental.

Generalidad. La amplitud de aplicación potencial de los componentes del programa.

Independencia del hardware. El grado en que el software es independiente del hardware sobre el que opera.

Instrumentación. El grado en que el programa muestra su propio funcionamiento e identifica errores que aparecen.

Modularidad. La independencia funcional de los componentes del programa.

Facilidad de Operación. La facilidad de operación de un programa.

Seguridad. La disponibilidad de mecanismos que controlen o protejan los programas o los datos.

Autodocumentación. El grado en que el código fuente proporciona documentación significativa.

Simplicidad. El grado en que un programa puede ser entendido sin dificultad.

Independencia del sistema de software. El grado en que el programa es independiente de características no estándar del lenguaje de programación, de las características del sistema operativo y de otras restricciones del entorno.

Facilidad de Traza. La posibilidad de rastrear la representación del diseño o de los componentes reales del programa hacia atrás, hacia los requisitos.

Formación. El grado en que el software ayuda para permitir que nuevos usuarios apliquen el sistema.

El esquema de graduación para estas métricas, propuesto por McCall [1977], va en una escala de 0 (bajo) a 10 (alto). La relación entre los factores de calidad del software y las métricas cualitativas de calidad se muestran en la tabla 2.1. El peso dado en cada métrica dependerá de los productos particulares, y de otros aspectos.

Métrica Calidad Software vs. Factor de Calidad	Corrección	Fiabilidad	Eficiencia	Integridad	Facilidad de mantenimiento	Flexibilidad	Facilidad de Prueba	Portabilidad	Reusabilidad	Facilidad de interoperación	Facilidad de uso
Facilidad de Auditoria				X			X				
Exactitud		X									
Norm. de las Comunicaciones										X	
Completitud	X										
Complejidad		X				X	X				
Concisión			X		X	X					
Consistencia	X	X			X	X					
Estandarización de los datos										X	
Tolerancia de Errores		X									
Eficiencia en la ejecución			X								
Facilidad de Expansión						X					
Generalidad						X		X	X	X	
Independencia del Hardware								X	X		
Instrumentación				X	X		X				
Modularidad		X			X	X	X	X	X	X	
Facilidad de Operación			X								X
Seguridad				X							
Autodocumentacón					X	X	X	X	X		
Simplicidad		X			X	X	X				
Indep. del sist. de software								X	X		
Facilidad de Traza	X										
Formación											X

Tabla 2.1 Factores y Métricas de Calidad.

2.4.4 MÉTRICAS CUANTITATIVAS DE LA CALIDAD DEL SOFTWARE

Algunas de la métricas cuantitativas de la calidad del software son:

1) Índices de calidad del software.

2) Medida de complejidad de McCabe.

3) Métricas de Halstead.

Índices de calidad del software.

El US Force Systems Command ha desarrollado una serie de indicadores de calidad del software basados en las características de diseño medibles para un programa de computadora. La Air Force utiliza información obtenida a partir del diseño

arquitectónico y de datos, para obtener un índice de calidad de la estructura del diseño, cuyo valor está entre 0 y 1.

El estándar de IEEE 982.1 sugiere un índice de madurez del software, el cual proporciona una indicación de la estabilidad de un producto de software (basada en los cambios que se producen en cada versión del producto).

Medida de complejidad de McCabe.

Esta medida propuesta por Thomas McCabe [McCabe 1976] se basa en la representación del flujo de control de un programa. Para describir el flujo de control se usa una red del programa (Fig. 2.6) [Preesman,1993]. Cada círculo representa una tarea de procesamiento (una o más sentencias de código); el flujo de control (ramificaciones) se representa mediante flechas correctivas.

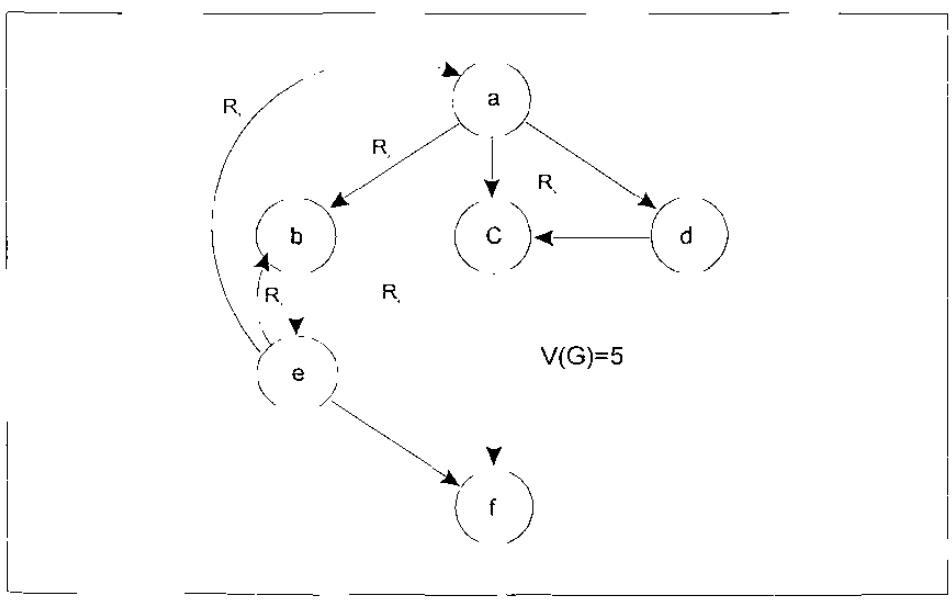


Fig. 2.6 Complejidad de la red de flujo de control.

McCabe [1976] define una medida de la complejidad del software que se basa en la complejidad ciclomática de la red del programa de un módulo. Una técnica que se puede usar para calcular la métrica de complejidad ciclomática, es la de determinar el número de regiones que forma la red en el plano.

Métricas de Halstead.

La teoría del Halstead sobre la ciencia del software asigna leyes cuantitativas al desarrollo de software de computadora. La teoría de Halstead se deriva de la suposición fundamental: "El cerebro humano sigue un conjunto de reglas más rígido (al desarrollar algoritmos) del que nunca ha tenido consciencia ..." [Halstead, 1977, p.15]. En esta teoría se usa un subconjunto de medidas básicas que se pueden obtener una vez que se generado el código fuente o estimar una vez que se terminado el diseño.

En el siguiente capítulo se aborda más a fondo la ciencia del software de Halstead

2.5 RESUMEN DEL CAPÍTULO

En este capítulo se ha presentado una breve explicación del lugar en que se ubican las métricas de Software, así como de los conceptos que nos conducen hacia ellas. En el siguiente capitulo se analizarán en forma más detallada las métricas de Halstead, la cuales son utilizadas en nuestro estudio para medir la calidad del software desarrollado

CAPÍTULO 3

MÉTRICAS DE HALSTEAD Y PREGUNTAS DE LA INVESTIGACIÓN

3.0 INTRODUCCIÓN

Este capítulo se divide en dos parte fundamentales. La primera parte trata la descripción explícita de las métricas de Halstead, en base a la teoría de la Ciencia de Software [Halstead,1977], como herramienta que permitirá evaluar la calidad del software. La segunda parte trata las preguntas e hipótesis de la investigación.

El presente capitulo se dirige de la siguiente forma. En la sección 3.1 se definen los conceptos básicos de la Ciencia del Software. En 3.2 se explica el estimador de longitud de un programa. En 3.3 se explica el estimador del volumen potencial de un programa. En 3.4 se explica el estimador del nivel/dificultad y se define el contenido de inteligencia de un programa. En 3.5 se explican el estimador del esfuerzo y del tiempo de programación de un programa. En 3.6 se define el nivel del lenguaje de un programa y se explica su estimador. En 3.7 se da una clasificación de los niveles de lenguaje para

diferentes lenguajes y sus distribuciones de frecuencia. En 3.8 se proponen las preguntas e hipótesis de la investigación. En 3.9 se hace un resumen del capítulo.

3.1 DEFINICIONES BÁSICAS

Un programa computacional se considera, en la Ciencia del Software, como una serie de partículas elementales que pueden ser clasificadas como operadores ú operandos. Esto se basa en el hecho de que todos los programas pueden ser reducidos en una sucesión de instrucciones de lenguaje máquina, cada una de las cuales contiene un operador y un número de direcciones de operandos. Todas las medidas de la ciencia del software están en función de la cantidad de estas partículas elementales. La métricas básicas se definen como:

n_1 = ***número de operadores diferentes*** *(1)*
n_2 = ***número de operandos diferentes*** *(2)*
N_1= ***número total de operadores*** *(3)*
N_2 = ***número total de operandos*** *(4)*

Generalmente, se considera operador cualquier símbolo en un programa que especifique una acción algorítmica, mientras que un símbolo usado para representar datos se considera un operando. La mayoría de las señales de puntuación se clasifican como operadores. El vocabulario de un programa, que consiste del número de las diferentes partículas elementales usadas para construir un programa, se define como:

$$n = n_1 + n_2 \qquad (5)$$

La longitud de un programa, en términos del número total de partículas elementales usado, se define como:

$$N = N_1 + N_2 \qquad (6)$$

Obsérvese que N está estrechamente relacionada con la tradicional medida de líneas de código (LOC) de la longitud de un programa. Para los programas en lenguaje máquina donde cada una de las líneas consiste de un operador y de un operando, se tiene que $N = 2 \times LOC$.

Se definen adicionalmente otras métricas usando estos términos básicos. Una métrica interesante para el tamaño del programa es la llamada volumen.

$$V = N \log_2 (n) \qquad (7)$$

La unidad de medida del volumen es la unidad común para tamaño, a saber, "bits". El volumen es el tamaño actual de un programa en una computadora si se utiliza una codificación binaria uniforme para el vocabulario.

El volumen también se puede interpretar como el número de comparaciones mentales que se necesitan para escribir un programa de longitud N, suponiendo que se usa un método de inserción binaria para seleccionar un miembro del vocabulario del tamaño n.

Como un algoritmo se puede implementar de diferentes maneras, pero en programas equivalentes, un programa que tiene el tamaño mínimo se dice que tiene el volumen potencial, el cual denotaremos por V^*.

Cualquier programa con un volumen V se considera que se implementa en el nivel del programa L, el cual se define por:

$$L = V^* / V \qquad (8)$$

El valor de L se encuentra en el intervalo semiabierto (0,1], donde $L = 1$ representa un programa escrito en el más alto nivel posible (i.e., con tamaño mínimo).

Lo inverso del nivel del programa se llama dificultad

$$D = 1 / L \qquad (9)$$

Cuando el volumen de una implementación de un programa crece, el nivel del programa decrece y la dificultad se incrementa. De este modo, las prácticas de programación tales como el uso redundante de operandos ó el error de usar frases de control de nivel más alto tenderán a incrementar el volumen así como la dificultad.

El esfuerzo que se requiere para implementar un programa de computadora se incrementa cuando el tamaño del programa crece. También toma más esfuerzo implementar un programa en un nivel más bajo (dificultad más alta) comparado con otro programa equivalente en un nivel más alto (dificultad más baja). Por ejemplo, toma más esfuerzo implementar un programa que despliegue una pantalla en PASCAL que implemenarlo en FOXPRO2. De ese modo. Se define el esfuerzo en la ciencia del software como:

$$E = V / L = D \ V \qquad (10)$$

La unidad de medida E es "discriminaciones binarias mentales elementales" [Halstead,1977].

3.2 ESTIMADOR DE LA LONGITUD DE UN PROGRAMA

La primera hipótesis de la ciencia del software es que la longitud de un programa bien estructurado solamente es función del número de operadores y operandos diferentes. Esta es llamada la ecuación de longitud, donde *Nest* es la longitud estimada del programa.

$$Nest = n_1 \ log_2 (n_1) + n_2 log_2 (n_2) \qquad (11)$$

Hay evidencia empírica de que *Nest* es un estimador aceptable de N cuando se aplica a un amplio rango de programas escritos en Fortran, Cobol, PL/S, DBASE III, FOXPRO2, C++, PROGRESS6 y ORACLE [Shen,1983], [Martínez,1994],

[Espinosa,1997] y [Navarro 1997] . Estos lenguajes son de los llamados de tercera y cuarta generación.

3.3 ESTIMADOR DEL VOLUMEN POTENCIAL DE UN PROGRAMA

Como se discutió antes, un programa que implementa un algoritmo en su forma más breve tiene el volumen potencial V^*. Si la función que se desea está ya definida en el lenguaje de programación o en su librería de subrutinas como un procedimiento, el volumen potencial se logra especificando el nombre del procedimiento y dando una lista de parámetros entrada/salida. El vocabulario de este programa consiste en dos operadores de n_2^* operandos. Uno de los operadores es el nombre del procedimiento, ya que define una acción; el otro operador es un símbolo de agrupamiento que se necesita para separar la lista de parámetros del nombre de procedimientos. De este modo

$$V^* = (2+n_2^*)\log_2(2+n_2^*) \qquad (12)$$

Donde $n2^*$ es el número de parámetros entrada/salida para un procedimiento. Esta fórmula, aunque es útil para muchos programas, no es aplicable universalmente, ya que hay programas que no tienen una lista explícita de parámetros entrada/salida. Por ejemplo, muchos de los programas escritos en FOXPRO2 o DBASE III, no traen consigo esta lista de parámetros.

3.4 ESTIMADOR DEL NIVEL/DIFICULTAD DE UN PROGRAMA Y DEFINICIÓN DEL CONTENIDO DE INTELIGENCIA DE UN PROGRAMA.

El nivel (8) de una implementación particular depende de la razón entre el volumen potencial y el volumen actual. Ya que el volumen potencial, usualmente, no está disponible, una fórmula alternativa que estima el nivel se define como:

$$Lest = 1/Dest = (2/n_1)(n_2/N_2) \qquad (13)$$

Un argumento intuitivo para esta fórmula es que la dificultad de la programación se incrementa si se introducen operadores extra (aumenta $n_1/2$) y si un operando se usa repetidamente (aumenta N_2 / n_2). Cada parámetro en (13) se puede obtener contando los operadores y operandos en un programa computacional. El volumen potencial V^*, entonces, se puede deducir usando (8) con L igual a $Lest$.

La ecuación (8) sugiere que para un algoritmo dado, diferentes implementaciones pudieran tener volúmenes y niveles diferentes, mientras que el producto de estos dos pudiera mantenerse constante. Esto es, el volumen potencial $V^*=L\ V$, depende solamente del algoritmo, no sobre las características de una implementación particular.

Cuando $Lest$ de (13) se utiliza para estimar L, el producto $(Lest)x(V)$ es llamado inteligencia I, esto es:

$$I = Lest\ V \qquad (14)$$

Se espera, también, que el contenido de inteligencia I se mantenga constante en diferentes implementaciones del mismo problema, ya que es un estimador de V^*.

3.5 ESTIMADOR DEL ESFUERZO Y DEL TIEMPO DE PROGRAMACIÓN DE UN PROGRAMA

Una de las mayores demandas por la ciencia del software es su capacidad para relacionar sus métricas básicas al tiempo común de implementación. El psicólogo, J. Stroud, sugirió que el humano es capaz de hacer un número limitado de discriminaciones elementales por segundo [Halstead,1977]. Stroud afirmó que este número S (ahora llamado número Stroud) está clasificado entre 5 y 20. Como el esfuerzo E tiene como unidad de medida "el número de discriminaciones elementales", el tiempo de programación T, en segundos, de un programa es :

$$T = E / S \qquad (15)$$

S está normalmente colocado en 18, ya que esto pareció dar el mejor resultado en los experimentos de Halstead comparando los tiempos de predicción con los tiempos de programación observados que incluyeron: tiempo de diseño, codificación y prueba.

De (10) tenemos que $E=V/L$, pero utilizando la estimación de L, ***Lest*** en (13), y V de (7) tenemos que el estimador del esfuerzo ***Eest*** es:

$$E\,est = (n_1\ N_2\ N\ \log_2(n)) / (2\ n_2) \qquad (16)$$

y por lo tanto, de (15) y (16), el tiempo estimado de programación es:

$$Test = (n_1 N_2 N \log_2(n)) / (36 n_2) \qquad (17)$$

La ciencia del software sostiene que esta fórmula puede ser usada para estimar el tiempo de programación cuando un problema dado es resuelto por un sólo programador hábil y concentrado que escribe un programa de un sólo módulo.

3.6 DEFINICIÓN DEL NIVEL DEL LENGUAJE DE PROGRAMACIÓN Y SU ESTIMADOR

La proliferación de lenguajes de programación sugiere la necesidad de una métrica que exprese el poder de un lenguaje. Halstead hipotetizó que si el lenguaje de programación se mantiene fijo, entonces V^* crece, L decrece de tal forma que el producto $L \times V^*$ se mantiene constante. Así, este producto llamado nivel del lenguaje (λ), se puede usar para caracterizar un lenguaje de programación.

$$\lambda = L\ V^* = L^2\ V \qquad (18)$$

sustituyendo, ***Lest*** por L de la ecuación (13) y V de la ecuación (7), tenemos que el estimador para el nivel del lenguaje λest es:

$$\lambda est = [\ (2 / n_1)\ (n_2 / N_2)]^2\ \ N\ \log_2(n) \qquad (19)$$

3.7 NIVELES DE LENGUAJE PARA DIFERENTES LENGUAJES Y SUS DISTRIBUCIONES DE FRECUENCIA

Analizando un número de programas diferentes escritos en lenguajes diferentes, se determinó los niveles de lenguaje para cada uno de ellos (Tabla 3.1) [Martínez,1994] [Espinosa,1997], [Navarro,1997] y [Baez-Ventura 1997]

Lenguaje	λ	σ^2
Inglés	2.16	0.74
FoxPro2	1.97	3.28
Progress	1.97	3.34
Dbase III	1.95	2.90
C++	1.84	2.94
PL/1	1.53	0.92
Algol 58	1.21	0.74
Fortran	1.14	0.81
Pilot	0.92	0.43
Assembly	0.88	0.42

Tabla 3.1 Media y Varianza del Nivel del Lenguaje

Estos valores promedio obedecen a la mayoría de las clasificaciones intuitivas de los programadores para estos lenguajes, pero todos ellos tienen grandes varianzas. Tales fluctuaciones en un valor fijo hipotetizado no son completamente inesperadas ya que el nivel del lenguaje no sólo depende del lenguaje en sí mismo, sino también de la naturaleza del problema que está siendo programado así como de la habilidad y estilo del programador.

En la tabla 3.2 se pueden ver las distribuciones de frecuencia de los niveles de lenguaje para diferentes lenguajes. [Halstead,1977], [Martínez,1994], [Espinosa,1997], [Navarro,1997] y [Baez-Casselyn,1997].

Intervalo	Comp	Pilot	Fort	Alg 58	PL/I	C++	DB III	PROG 6	FxPro2	Ing
(0,0.99]	71	78	57	43	43	44.35	26.92	35.50	34.19	0
(1,1.99]	28	14	36	50	35	22.91	42.31	31.50	26.50	40
(2,2.99]	0	7	0	0	21	9.65	15.38	15.20	20.51	44
(3,3.99]	0	0	7	7	7	13.42	7.69	5.50	10.26	17
(4,4.99]	0	0	0	0	0	3.27	0.00	3.90	5.31	0
(5,5.99]	0	0	0	0	0	4.41	3.85	4.40	0.85	0
(6,6.99]	0	0	0	0	0	0.65	0.00	1.60	0.00	0
(7,7.99]	0	0	0	0	0	0.32	0.00	1.00	0.85	0
(8,8.99]	0	0	0	0	0	0.49	3.85	0.40	0.85	0
(9,9.99]	0	0	0	0	0	0.00	0.00	0.50	0.00	0
(10,10.99]	0	0	0	0	0	0.16	0.00	0.10	0.00	0
(11,11.99]	0	0	0	0	0	0.16	0.00	0.20	0.00	0
(12,12.99]	0	0	0	0	0	0.00	0.00	0.20	0.00	0
(13,13.99]	0	0	0	0	0	0.00	0.00	0.00	0.00	0
(14,14.99]	0	0	0	0	0	0.16	0.00	0.00	0.85	0

Tabla 3.2 Distribuciones de frecuencia de los niveles de lenguaje para diferentes lenguajes

3.8 PROPUESTA DE PREGUNTAS E HIPÓTESIS DE INVESTIGACIÓN.

En el área de educación se ha demostrado que los estudiantes que han recibido una enseñanza en las métricas de software producen programas que exhiben menos complejidad, requieren menos tiempo de codificación y prueba, y además, es más fácil de darles mantenimiento [Bowman,1990].

En consecuencia, la primer pregunta de la investigación es: Si se imparte un curso de métricas de software a un grupo de estudiantes de Sistemas computacionales que están involucrados en un proyecto de desarrollo de software donde utilizan el lenguaje FOXPRO2.6 para Windows, ¿puede contribuir para que ellos, posteriormente, generen código de mejor calidad tal que permita su facilidad de mantenimiento?.

Para poder dar respuesta a la pregunta que anteriormente se formuló, se contará con dos grupos de seis estudiantes; un grupo será el experimental y otro de control. Al grupo experimental se le impartirá un curso de métricas de software con una duración de seis horas. Después, los estudiantes harán 10 programas cada uno; los programas serán los mismos. A esos programas se les aplicarán las métricas de Halstead y,

finalmente, se compararán los resultados del grupo de control contra los resultados del grupo experimental. De aquí surgen las hipótesis que se proponen a continuación:

H1: La media del volumen del programa de los programas hechos por el grupo experimental será más bajo que la media del volumen del programa de los programas hechos por el grupo de control.

H2: La media de nivel del programa de los programas hechos por el grupo experimental será más alta que la media de nivel del programa de los programas hechos por el grupo de control.

H3: La media del contenido de inteligencia de los programas hechos por el grupo experimental será más alta que la media del contenido de inteligencia de los programas hechos por el grupo de control.

H4: La media del esfuerzo de programación de los programas hechos por el grupo experimental será más bajo que la media del esfuerzo de programación de los programas hechos por el grupo control.

H5: La media del tiempo estimado de programación de los programas hechos por el grupo experimental será más bajo que la media del tiempo estimado de programación de los programas hechos por el grupo de control.

H6: La media del nivel del lenguaje de los programas hechos por el grupo experimental será más alta que la media del nivel del lenguaje de los programas hechos por el grupo de control.

3.9 RESUMEN DEL CAPÍTULO

En este capítulo se presentó el marco teórico en el que se basa esta investigación, y se propusieron las preguntas e hipótesis de la investigación. En el siguiente capítulo discutiremos el enfoque que se tomó para investigar las hipótesis propuestas.

CAPÍTULO 4

METODOLOGÍA DE LA INVESTIGACIÓN

4.0 INTRODUCCIÓN

En el capítulo anterior se discutió el marco teórico de la investigación y se propusieron las preguntas e hipótesis de la investigación. En este capitulo se discute la metodología empleada para desarrollarla.

El presente capítulo se dirige de la siguiente forma. En la sección 4.1 se da a conocer el diseño de la investigación. En la 4.2 se definieron las variables de la investigación. En la 4.3 se da a conocer el instrumento de medición. En la 4.4 se da a conocer cómo se formaron el grupo experimental y de control. En la 4.5 se dan a conocer las razones de la elección del lenguaje de programación. En la 4.6 se da a conocer como se obtuvo la equivalencia del grupo experimental con el grupo de control a través del emparejamiento. En la 4.7 se da a conocer cómo se realizó la recolección de datos. En la 4.8 se dan a conocer las hipótesis de causalidad bivariada. En la 4.9 se da a conocer la

prueba estadística que se utilizará para comprobar las hipótesis y en la 4.10 se da un resumen del capítulo.

4.1 DISEÑO DE INVESTIGACIÓN

El tipo de estudio realizado en la presente tesis es del tipo explicativo[1], donde se establecieron una variable dependiente y una independiente. Luego, se formó un grupo experimental y un grupo de control, el grupo experimental fue expuesto a la presencia de la variable independiente. La equivalencia entre los grupos fue realizado por el método de emparejamiento[2].

De esta manera, nos disponemos a explicar por qué el conocimiento sobre las métricas de software influye para que genere software de mejor calidad.

Para que un estudio explicativo se pueda dar, es necesario que antes se haya establecido causalidad entre las variables que intervienen, misma que ya ha sido demostrada por Bowman [1990].

Por otro lado, un estudio explicativo establece causalidad, por tanto se hipotetiza que hay una causalidad entre las variables que intervienen en este estudio, donde la causa es la enseñanza de las métricas de software y el efecto es una mejor calidad en el software desarrollado.

4.2 DEFINICIÓN DE VARIABLES

En este caso particular hay una variable dependiente y una independiente.

- La variable independiente es el curso de métricas de software que se le impartió al grupo experimental.
- La variable dependiente es la Calidad de los Programas, que se va a medir en ambos grupos: el experimental y el de control.

[1] Explicativo: Se centra en explicar por que dos o más variables están relacionadas.
[2] El emparejamiento intenta hacer los grupos inicialmente equivalentes (Experimental y Control).

4.2.1 VARIABLE INDEPENDIENTE.

La variable independiente es el curso de métricas de software, el cual se impartió al grupo experimental en un lapso de seis horas. Los temas incluyeron los capítulos 1,2,3,20 del libro: Ingeniería de Software: un enfoque práctico de Roger S. Pressman, 1993, Tercera Edición; se incluyó el tema de la Ciencia del Software de Halstead, que es el capítulo 3 de esta tesis. Además, se incluyó un programa de prueba, en dos versiones, donde se mostraba los resultados después de someterlos a las métricas de Halstead. Para ver el contenido especifico de este curso favor de remitirse al Anexo E de esta tesis (en el disquete).

4.2.2 VARIABLE DEPENDIENTE.

La variable dependiente es la calidad de un programa, la cual se entiende como la facilidad de mantenimiento, es decir, es fácil de corregir, adaptar y/o mejorar.

4.3 INSTRUMENTO DE MEDICIÓN

La variable dependiente no se manipula, sólo se mide para ver el efecto de la manipulación de la variable independiente sobre ella , entonces lo que se mide es el efecto de la variable independiente sobre la variable dependiente [Hernández,1995].

La calidad de los programas fue medida con el analizador de código que fue desarrollado por Martínez [1994]. El analizador antes mencionado fue utilizado para programas de FOXPRO2; por lo cual no necesitó actualización puesto que las diferencias existentes entre los mandatos de la versión 2 con la 2.5 para DOS y Windows son escasas pero importantes sobre todo en lo referente al diálogo entre ventanas [García-Badell,1997] y la diferencia con la versión 2.6 son todavía menores. Se hizo una prueba para validar si el instrumento era confiable. La prueba de validación se constató al someter varios programas hechos en FOXPRO2.6 para Windows. Una muestra para

validación se encuentra en el curso de métricas de software del Anexo E (el programa de prueba en dos versiones).

4.4 GRUPO EXPERIMENTAL Y DE CONTROL

Los grupos estuvieron compuestos por seis alumnos cada uno. Todos los participantes fueron alumnos en la materia de programación III (FOXPRO2.6 para Windows) en el semestre de febrero-julio de 1998. El reclutamiento de los alumnos se gestó desde noviembre de 1997, a través de una convocatoria abierta cuyo requisito fue: ser alumno que iba a cursar la materia de Programación III de la carrera de Ingeniero Administrador de Sistemas.

En el experimento se prefirió alumnos de la carrera de I.A.S. dado que los alumnos más interesados serían los de esta carrera, y así todos los participantes tendrían los mismos intereses. Antes de ingresar al semestre normal en febrero del '98, se les impartió, en el mes de enero del '98, un curso de 20 horas de FOXPRO2.6 para Windows. La intención de impartirles dicho curso, fue el de asegurarse que al finalizar el semestre, estos participantes tuvieran una exposición más alta al lenguaje de programación que se utilizaría en el experimento que el resto de compañeros.

Por tanto, teniendo en mente su nivel académico, podemos decir que eran buenos programadores en FOXPRO2.6 para Windows, con lo que eliminamos posibles argumentaciones de invalidación interna, por lo menos en este sentido.

4.5 SELECCIÓN DEL LENGUAJE DE PROGRAMACIÓN.

Se eligió el lenguaje de programación FOXPRO2.6 para Windows por las siguientes razones:

1) En el ambiente regional, es uno de los dos lenguajes de programación (FOXPRO2) más utilizados en la mediana empresa [Crespo,1992].

2) Porque en 1997 fue el lenguaje de programación más solicitado en los anuncios de oferta de empleos en el periódico EL NORTE y en la bolsa de trabajo de la F.I.M.E.-U.A.N.L. para la carrera de I.A.S.[Gutiérrez,1998].
3) Porque ya existía una investigación anterior que había demostrado que las métricas de Halstead eran válidas para el lenguaje FOXPRO2 y se había desarrollado el analizador de código [Martínez,1994].
4) Porque se llegó a la conclusión que el analizador de código de la investigación anterior era válido para FOXPRO2.6 para Windows [García-Badell,1997]
5) Porque FOXPRO2.6 para Windows es impartido actualmente en la materia de Programación III en la F.I.M.E.-U.A.N.L.

4.6 EQUIVALENCIA DE LOS GRUPOS (EMPAREJAMIENTO).

En un estudio explicativo se debe buscar la validez interna y esto se logra mediante:

1) Varios grupos de comparación (dos como mínimo)
2) Equivalencia de los grupos en todo, excepto la manipulación de la variable o las variables independientes.

Es decir, para tener control no basta con tener dos o más grupos, sino que éstos deben ser similares en todo excepto la manipulación de la variable independiente [Hernández,1995] . Para lograr lo anterior, se recurrió al método de emparejamiento el cual consiste en igualar a los grupos en relación con alguna variable, que se piensa puede influir en forma decisiva en la variable dependiente o variables independientes.

El emparejamiento en esta investigación se realizó a través de la calificación que los alumnos obtuvieron en la materia, dado que es una variable que evalúa su habilidad para programar. Los grupos quedaron como a continuación se muestra. (Tabla 4.1).

Grupo Experimental			Grupo de Control	
Alumno	Calificación		Alumno	Calificación
A	70		A	70
B	73		B	81
C	85		C	81
D	85		D	85
E	88		E	89
F	93		F	90
Suma	494		Suma	496
Promedio	82.33		Promedio	82.67

Tabla 4.1 Grupos Equivalentes.

4.7 RECOLECCIÓN DE DATOS.

Los programas que cada uno de los participantes desarrollaron fueron los mismos diez, esto es, en total hablamos de 60 algoritmos (codificaciones) del grupo de control y 60 del grupo experimental. Los programas fueron reportes de un Sistema de Administración Comercial (Anexo A, en el disquete) que viene en un curso de Progress6. Se eligió este sistema porque es un sistema hecho por personas expertas con fines educativos pues ya se tiene probado que sirve para un curso de capacitación y además ya tiene el diagrama entidad-relación.

Antes de hacer los programas que sirvieron para la investigación, los participantes hicieron los programas de "altas" para este Sistema Comercial de Administración, por lo que podemos decir que los alumnos tuvieron una buena exposición al sistema del cual se hicieron los reportes finales (programas). Esto ayudó a la validación interna en cuanto al conocimiento del sistema en cuestión.

Los programas que sirvieron para la investigación fueron hechos después de concluir el semestre normal, en julio de 1998. Es bueno destacar que todos los participantes aprobaron la materia de programación III.

4.8 HIPÓTESIS DE CAUSALIDAD BIVARIADAS

Las hipótesis de causalidad no solamente afirman las relaciones entre dos o más variables y cómo se dan dichas relaciones, sino que además proponen un "sentido de entendimiento" entre ellas [Hernández,1995].

En la presente investigación se desea saber si impartir un curso de métricas de software a los estudiantes de Sistemas Computacionales contribuye para que ellos, posteriormente, generen código de mejor calidad tal que permita su facilidad de mantenimiento.

Por lo tanto, se han generado seis hipótesis de diferencia de medias entre grupos con un cierto sentido de entendimiento. La explicación, en caso de encontrarse que son ciertas, es que el curso de métricas del software sí contribuye para que se genere código de mejor calidad permitiendo su facilidad de mantenimiento.

Las hipótesis son:

H1: La media del volumen del programa de los programas hechos por el grupo experimental será más bajo que la media del volumen del programa de los programas hechos por el grupo de control.

H2: La media de nivel del programa de los programas hechos por el grupo experimental será más alta que la media de nivel del programa de los programas hechos por el grupo de control.

H3: La media del contenido de inteligencia de los programas hechos por el grupo experimental será más alta que la media del contenido de inteligencia de los programas hechos por el grupo de control.

H4: La media del esfuerzo de programación de los programas hechos por el grupo experimental será más bajo que la media del esfuerzo de programación de los programas hechos por el grupo control.

H5: La media del tiempo estimado de programación de los programas hechos por el grupo experimental será mas bajo que la media del tiempo estimado de programación de los programas hechos por el grupo de control.

H6: La media del nivel del lenguaje de los programas hechos por el grupo experimental será más alta que la media del nivel del lenguaje de los programas hechos por el grupo de control.

4.9 PRUEBA ESTADÍSTICA

Las hipótesis anteriormente formuladas son llamadas de diferencia entre grupos y se formulan en investigaciones dirigidas a comparar grupos. Ahora bien, si las hipótesis de diferencias entre grupos, además de establecer tales diferencias explican el por qué de las diferencias (las causas o razones de estas), entonces son hipótesis de estudios explicativos [Hernández,1995].

La prueba estadística que suele utilizarse para comparar las medias de dos grupos con muestras grandes, mayor que 30, es la prueba Z, que además indica la dirección y el grado en que un valor individual obtenido se aleja de la media en una escala de unidades de desviación estándar.

La prueba Z tiene la hipótesis nula y la hipótesis alternativa [Mendenhall,1986] donde:

Ho: $\mu_1 = \mu_2$ *y* ***Ha:*** $\{\mu_1 > \mu_2,\ \mu_1 < \mu_2$ *ó* $\mu_1 \neq \mu_2\}$

- $\mu_1 > \mu_2$ (alternativa de cola superior)
- $\mu_1 < \mu_2$ (alternativa de cola superior)
- $\mu_1 \neq \mu_2$ (alternativa de dos colas)

Estadístico de Prueba: $Z = \mu\hat{} - \mu / \sigma_{\mu}$

Región de Rechazo de la ***Ho*** $\{$ $Z > Z\alpha$ cola superior, $Z < -Z\alpha$ cola inferior y $|Z| > Z\alpha$ dos colas.

Donde α es el nivel de significancia asociada con una prueba. La selección del α es arbitrario, aunque se recomienda valores pequeños. En este caso se ha elegido valores de 0.10 y 0.20. El valor más usado para trabajos de ingeniería es 0.05, pero dado que el software es un elemento lógico y no físico, y lo que estamos midiendo son procesos muy complejos de la mente humana, se considera que encontrar una significancia igual o mayor al 80% es prueba suficiente para aceptar las hipótesis de investigación (Ha).

4.10 RESUMEN DEL CAPÍTULO

En este capítulo discutimos acerca del tipo de estudio realizado y el diseño de la investigación. Se definieron las variables de la investigación. Se discutió acerca del instrumento de medición. Se dieron a conocer la razones para elegir el lenguaje de programación. Se discutió el proceso de hacer equivalentes el grupo de control y el grupo experimental. Se dio a conocer como se obtuvo la muestra a ser examinada. Se propusieron las hipótesis de esta tesis y finalmente se definió la prueba estadística que se utilizará.

CAPÍTULO 5

ANÁLISIS DE DATOS

5.0 INTRODUCCIÓN

Este capítulo presenta el análisis de datos que se efectuó para responder a las hipótesis de investigación. El capítulo se dirige de la siguiente forma. En la sección 5.1 se hace una revisión de la exactitud de los datos. En la 5.2 se presentan algunas abreviaturas utilizadas en este capítulo. En la 5.3 se muestran las tabla generales de datos para ambos grupos. En la 5.4 se dan las estadísticas descriptivas. En la 5.5 se hace el análisis de la primera hipótesis de la investigación. En la 5.6 se hace el análisis de la segunda hipótesis de la investigación. En la 5.7 se hace el análisis de la tercera hipótesis de la investigación. En la 5.8 se hace el análisis de la cuarta hipótesis de la investigación. En la 5.9 se hace el análisis de la quinta hipótesis de la investigación. En la 5.10 se hace el análisis de la sexta hipótesis de la investigación, y en la 5.11 se hace un resumen del capítulo.

5.1 REVISIÓN DE LA EXACTITUD DE LOS DATOS

La flexibilidad del lenguaje Foxpro2.6 para Windows permite al programador escribir las primeras cuatro letras (al menos) del comando para obtener la ejecución

completa del mismo. Por ejemplo, para ejecutar el comando BROWSE, el programador puede escribir solamente BROW. Si esto fuera así el analizador de código contabilizaría el comando BROW como operando y no como operador[Martínez,1994]. Para evitar este error se les pidió a los programadores que escribieran el nombre del comando.

También, con el objetivo de que el analizador hiciera el conteo correcto, se le pidió a los programadores que escribieran las instrucciones de FoxPro2.6 con letra mayúscula y que los nombres de los identificadores (variables, campos y procedimientos o funciones definidas por el usuario) los escribieran con letra minúscula.

Con todo lo anterior, se buscó eliminar inexactitud de los datos en caso de que estuviera mal escrito el programa fuente.

Por otro lado, se debe reconocer, que no es posible eliminar todas las impurezas a los programas porque el número de operadores que podemos formar dentro del leguaje es demasiado grande. Por ejemplo, sólo el comando BROWSE tiene al menos 16 subcomandos, los cual significa alrededor de 100 billones de forma distintas de construir el comando BROWSE [Pinter,1992]. Por lo tanto, es prácticamente imposible escribir cada uno de los comandos que podemos formar en FOXPRO2.6 para Windows [Martínez,1994].

La exactitud de los datos también depende del número total de operadores que estamos considerando. En efecto, si un operador no se encuentra en la lista de operadores asignada, éste será considerado como un operando. Sin embargo, se trató de eliminar este problema dando una lista de operadores lo suficientemente grande para que involucrara a todos los comandos. Además, en el conteo que se lleva a cabo en el analizador se trata también de eliminar este problema [Martínez,1994].

Por último, después de analizar los programas, en ninguno se suscitó el problema anteriormente formulado, es decir, ningún operador fue contabilizado como operando o viceversa.

5.2 ABREVIATURAS DEL CAPÍTULO

En este capítulo utilizaremos algunas abreviaturas:

- *GE*. Grupo Experimental
- *GC*. Grupo de Control
- *Progr*. Programador que hizo los programas.
- *#Prog*. Número de Programa del 1 al 10 solamente.
- *LOC*. Líneas de Código.
- *n1*. Número de Operadores Diferentes.
- *n2*. Número de Operandos Diferentes.
- *N1*. Total de Operadores.
- *N2*. Total de Operandos.
- *N*. Longitud del Programa
- *N^*. Longitud Estimada del Programa.
- *Nivel Prog*. Nivel del Programa
- *Cont. Int*. Contenido de Inteligencia.
- *Tpo. Est. Prog*. Tiempo Estimado de Programación.
- *Nivel Leng*. Nivel del Lenguaje

5.3 TABLAS GENERALES DE DATOS DE LOS *GE*. Y *GC*.

A continuación mostraremos las tablas generales para el GE. y GC. donde se muestra cada uno de los rubros medidos por el analizador de código fuente.

En la tabla 5.1 se muestran los datos de los programas para el *GE*. ordenado por programador.

En la tabla 5.2 se muestran los datos de los programas para el *GC*. ordenado por programador.

En la tabla 5.3 se muestran los datos de los programas para el *GE*. ordenado por *N*.

En la tabla 5.4 se muestran los datos de los programas para el *GC*. ordenado por *N*.

Grupo Experimental															
Progr	#Prog	LOC	n1	n2	N1	N2	n	N	N^	Volumen	Nivel Prog	Cont. Int	Esfuerzo	Tpo. Est. Prog.	Nivel Leng
A	1	31	26	36	101	74	62	175	308.329	1,041.948	0.0370	38.993	27,844.137	1,546.897	1.459
A	2	49	39	44	129	94	83	223	446.346	1,421.634	0.0240	34.125	59,223.971	3,290.221	0.819
A	3	71	37	67	227	162	104	389	599.178	2,606.471	0.0220	58.269	116,590.951	6,477.275	1.303
A	4	51	38	39	135	103	77	238	405.552	1,491.495	0.0200	29.723	74,842.464	4,157.915	0.592
A	5	37	33	30	98	68	63	166	313.672	992.228	0.0270	26.53	37,109.345	2,061.630	0.709
A	6	39	34	34	120	77	68	197	345.947	1,199.230	0.0260	31.149	46,170.362	2,565.020	0.809
A	7	39	31	32	99	66	63	165	313.58	986.251	0.0310	30.85	31,529.218	1,751.623	0.965
A	8	35	36	25	110	73	61	183	302.214	1,085.325	0.0190	20.649	57,044.678	3,169.149	0.393
A	9	96	27	70	337	237	97	574	557.432	3,788.350	0.0220	82.883	173,154.653	9,619.703	1.813
A	10	45	37	34	117	83	71	200	365.724	1,229.949	0.0220	27.234	55,546.686	3,085.927	0.603
B	1	29	24	35	99	72	59	171	289.564	1,005.932	0.0410	40.750	24,832.149	1,379.564	1.651
B	2	43	37	40	122	85	77	207	405.627	1,297.225	0.0250	32.998	50,997.150	2,833.175	0.839
B	3	75	38	65	231	164	103	395	590.875	2,641.168	0.0210	55.095	126,613.517	7,034.084	1.149
B	4	48	35	39	133	102	74	235	385.656	1,459.222	0.0220	31.882	66,787.447	3,710.414	0.697
B	5	36	31	30	98	69	61	167	300.787	990.433	0.0280	27.782	35,308.941	1,961.608	0.779
B	6	38	33	35	112	74	68	186	345.990	1,132.268	0.0290	32.456	39,499.981	2,194.433	0.930
B	7	38	30	34	97	63	64	160	320.180	960.000	0.0360	34.540	26,682.353	1,482.353	1.243
B	8	35	36	27	104	70	63	174	314.499	1,040.047	0.0210	22.287	48,535.513	2,696.417	0.478
B	9	106	26	70	363	272	96	635	551.261	4,181.451	0.0200	82.778	211,223.020	11,734.612	1.639
B	10	40	33	34	102	76	67	178	339.439	1,079.764	0.0270	29.276	39,824.232	2,212.457	0.794
C	1	40	23	34	116	95	57	211	277.016	1,230.740	0.0310	38.302	39,546.565	2,197.031	1.192
C	2	33	31	31	86	64	62	150	307.160	893.129	0.0310	27.910	28,580.142	1,587.786	0.872
C	3	67	35	63	228	154	98	380	556.094	2,513.590	0.0230	58.759	107,525.783	5,973.655	1.374
C	4	58	33	44	167	131	77	298	406.680	1,867.502	0.0200	38.015	91,741.055	5,096.725	0.774
C	5	30	28	24	84	54	52	138	244.645	786.661	0.0320	24.973	24,779.811	1,376.656	0.793
C	6	41	27	38	133	84	65	217	327.803	1,306.854	0.0340	43.792	38,999.269	2,166.626	1.467
C	7	24	25	23	52	38	48	90	220.138	502.647	0.0480	24.339	10,380.746	576.708	1.179
C	8	38	30	32	125	83	62	208	307.207	1,238.473	0.0260	31.832	48,164.334	2,676.907	0.818
C	9	105	22	83	341	271	105	612	627.236	4,109.118	0.0280	114.410	147,581.705	8,198.984	3.186
C	10	44	31	31	113	82	62	195	307.160	1,161.068	0.0240	28.319	47,603.800	2,644.656	0.691
D	1	32	25	35	95	71	60	166	295.621	980.544	0.039	38.669	24,863.790	1,381.322	1.525
D	2	28	31	30	77	49	61	126	300.787	747.273	0.039	29.517	18,918.459	1,051.026	1.166
D	3	60	31	66	217	139	97	356	552.510	2,349.569	0.031	71.976	76,699.187	4,261.066	2.205
D	4	46	32	41	124	94	73	218	379.660	1,349.382	0.027	36.785	49,499.272	2,749.960	1.003
D	5	33	29	23	86	53	52	139	244.923	792.361	0.030	23.714	26,475.197	1,470.844	0.710
D	6	38	27	34	124	70	61	194	301.356	1,150.563	0.036	41.396	31,978.885	1,776.605	1.489
D	7	24	24	26	52	40	50	92	232.251	519.235	0.054	28.125	9,585.873	532.548	1.523
D	8	57	31	43	193	116	74	309	386.909	1,918.721	0.024	45.887	80,229.314	4,457.184	1.097
D	9	82	25	61	321	218	86	539	477.871	3,463.757	0.022	77.537	154,733.394	8,596.300	1.736
D	10	38	30	32	95	65	62	160	307.207	952.671	0.033	31.267	29,026.707	1,612.595	1.026
E	1	28	23	31	90	69	54	159	257.622	915.027	0.0390	35.748	23,421.742	1,301.208	1.397
E	2	35	29	37	107	73	66	180	333.631	1,087.991	0.0350	38.031	31,125.362	1,729.187	1.329

Tabla 5.1 Datos Generales GE ordenado por programador

E	3	71	30	67	239	163	97	402	553.635	2,653.165	0.02700	72.704	96,820.721	5,378.929	1.992
E	4	55	32	40	138	111	72	249	372.877	1,536.311	0.0230	34.602	68,212.223	3,789.568	0.779
E	5	39	30	30	96	75	60	171	294.413	1,010.078	0.0270	26.935	37,877.936	2,104.330	0.718
E	6	49	25	42	162	97	67	259	342.574	1,571.117	0.0350	54.422	45,356.654	2,519.814	1.885
E	7	31	26	32	81	59	58	140	282.211	820.117	0.0420	34.216	19,657.187	1,092.066	1.428
E	8	44	30	41	141	91	71	232	366.866	1,426.741	0.0300	42.855	47,500.047	2,638.891	1.287
E	9	91	23	69	354	245	92	599	525.530	3,907.614	0.0240	95.697	159,560.889	8,864.494	2.344
E	10	44	32	28	103	79	60	182	294.606	1,075.054	0.0220	23.814	48,531.013	2,696.167	0.528
F	1	34	27	36	113	86	63	199	314.499	1,189.479	0.0310	36.883	38,360.688	2,131.149	1.144
F	2	38	30	36	103	73	68	176	346.628	1,071.393	0.0350	37.181	30,873.048	1,715.169	1.290
F	3	73	35	66	227	179	101	406	578.455	2,703.234	0.0210	56.956	128,301.213	7,127.845	1.200
F	4	63	31	38	159	142	69	301	353.001	1,838.666	0.0170	31.744	106,497.462	5,916.526	0.548
F	5	40	30	28	102	76	58	178	281.813	1,042.721	0.0250	25.611	42,453.625	2,358.535	0.629
F	6	39	28	36	117	82	64	199	320.723	1,194.000	0.0310	37.443	38,075.333	2,115.296	1.174
F	7	30	25	29	72	50	54	122	256.978	702.096	0.0460	32.577	15,131.385	840.633	1.512
F	8	43	32	36	123	94	68	217	346.117	1,320.979	0.0240	31.619	55,187.585	3,065.977	0.757
F	9	67	22	67	258	194	89	452	504.535	2,927.032	0.0310	91.898	93,228.138	5,179.341	2.885
F	10	43	30	35	115	83	65	198	326.732	1,192.429	0.0280	33.522	42,416.397	2,356.466	0.942

Tabla 5.1 Datos Generales GE ordenado por programador cont...

GRUPO DE CONTROL															
Progr	#Prog	LOC	n1	n2	N1	N2	n	N	N^	Volumen	Nivel Prog	Cont. Int.	Esfuerzo	Tpo. Est. Prog.	Nivel Leng
A	1	45	30	46	126	100	76	226	401.291	1,412.032	0.0310	43.302	46,044.509	2,558.028	1.328
A	2	48	38	43	117	68	81	185	432.751	1,172.872	0.0330	39.035	35,240.720	1,957.818	1.299
A	3	82	39	65	218	152	104	370	597.585	2,479.163	0.0220	54.368	113,049.819	6,280.545	1.192
A	4	76	36	54	167	139	90	306	496.881	1,986.507	0.0220	42.874	92,041.493	5,113.416	0.925
A	5	41	29	27	99	64	56	163	269.263	946.599	0.0290	27.541	32,534.953	1,807.497	0.801
A	6	45	30	42	124	72	72	196	373.684	1,209.305	0.0390	47.029	31,096.422	1,727.579	1.829
A	7	39	33	36	88	58	69	146	352.582	891.845	0.0380	33.549	23,708.202	1,317.122	1.262
A	8	46	35	42	125	79	77	204	406.002	1,278.424	0.0300	38.838	42,081.472	2,337.860	1.180
A	9	108	28	74	367	275	102	642	594.105	4,283.697	0.0190	82.336	222,868.023	12,381.557	1.583
A	10	59	35	42	126	91	77	217	406.002	1,359.893	0.0260	35.865	51,562.597	2,864.589	0.946
B	1	29	24	35	99	72	59	171	289.564	1,005.932	0.0410	40.750	24,832.149	1,379.564	1.651
B	2	47	36	41	128	92	77	220	405.777	1,378.693	0.0250	34.134	55,685.748	3,093.653	0.845
B	3	75	38	63	231	164	101	395	575.990	2,629.994	0.0200	53.174	130,080.315	7,226.684	1.075
B	4	47	35	38	132	100	73	232	378.946	1,436.039	0.0220	31.183	66,133.389	3,674.077	0.677
B	5	36	31	30	98	69	61	167	300.787	990.433	0.0280	27.782	35,308.941	1,961.608	0.779
B	6	38	33	34	112	74	67	186	339.439	1,128.293	0.0280	31.418	40,518.978	2,251.054	0.875
B	7	38	30	29	97	63	59	160	286.088	941.223	0.0310	28.884	30,670.884	1,703.938	0.886
B	8	35	36	26	104	70	62	174	308.329	1,036.030	0.0210	21.378	50,207.615	2,789.312	0.441
B	9	106	26	70	363	272	96	635	551.261	4,181.451	0.0200	82.778	211,223.020	11,734.612	1.639
B	10	40	34	32	103	77	66	180	332.974	1,087.991	0.0240	26.597	44,505.629	2,472.535	0.650
C	1	40	23	34	116	95	57	211	277.016	1,230.740	0.0310	38.302	39,546.565	2,197.031	1.192
C	2	33	31	31	87	65	62	152	307.160	905.038	0.0310	27.847	29,413.730	1,634.096	0.857
C	3	67	35	63	226	154	98	380	556.094	2,513.590	0.0230	58.759	107,525.783	5,973.655	1.374
C	4	58	33	44	167	131	77	298	406.680	1,867.502	0.0200	38.015	91,741.055	5,096.725	0.774
C	5	31	28	24	85	56	52	141	244.645	803.762	0.0310	24.605	26,256.225	1,458.679	0.753
C	6	41	27	38	133	84	65	217	327.803	1,306.954	0.0340	43.792	38,999.269	2,166.626	1.467
C	7	24	25	23	52	38	48	90	220.138	502.647	0.0480	24.339	10,380.746	576.708	1.179
C	8	38	30	32	125	83	62	208	307.207	1,238.473	0.0260	31.832	48,184.334	2,676.907	0.818
C	9	105	22	83	341	271	105	612	627.236	4,109.118	0.0280	114.410	147,581.705	8,198.984	3.186
C	10	44	31	31	113	82	62	195	307.160	1,161.068	0.0240	28.319	47,603.800	2,644.656	0.691
D	1	46	31	37	139	107	68	246	346.330	1,497.516	0.0220	33.409	67,125.137	3,729.174	0.745
D	2	28	31	29	79	51	60	130	294.462	767.896	0.0370	28.171	20,931.780	1,162.877	1.033
D	3	83	38	78	246	189	116	435	689.683	2,983.222	0.0220	64.798	137,342.939	7,630.163	1.407
D	4	73	39	49	196	156	88	352	481.251	2,273.720	0.0160	36.625	141,156.245	7,842.014	0.590
D	5	46	25	23	104	76	48	180	220.138	1,005.293	0.0240	24.339	41,522.982	2,306.832	0.589
D	6	63	33	58	200	129	91	329	506.228	2,141.064	0.0270	58.342	78,573.373	4,365.187	1.590
D	7	43	21	22	63	68	43	131	190.346	710.841	0.0310	21.903	23,070.011	1,281.011	0.675
D	8	55	38	51	172	118	89	290	488.715	1,877.963	0.0230	42.719	82,556.713	4,586.484	0.972
D	9	80	21	59	320	227	80	547	493.315	3,458.095	0.0250	85.600	139,701.163	7,761.176	2.119
D	10	58	36	39	139	103	75	242	392.248	1,507.374	0.0210	31.709	71,658.247	3,981.014	0.667
E	1	38	30	34	115	91	64	206	320.180	1,236.000	0.0250	30.787	49,621.765	2,756.765	0.767
E	2	43	38	31	94	57	69	151	353.001	922.387	0.0290	26.403	32,224.043	1,790.225	0.756

Tabla 5.2 Datos Generales GC ordenado por programador

E	3	91	40	66	229	175	106	404	611.807	2,718.080	0.0190	51.255	144,140.599	8,007.811	0.967
E	4	73	38	45	159	133	83	292	446.555	1,861.512	0.0180	33.149	104,534.213	5,807.456	0.590
E	5	46	32	30	109	78	62	187	307.207	1,113.435	0.0240	26.765	46,318.884	2,573.271	0.643
E	6	56	35	41	139	97	76	236	399.185	1,474.511	0.0240	35.614	61,048.347	3,391.575	0.860
E	7	27	30	26	62	44	56	106	269.418	615.580	0.0330	24.250	15,626.252	868.125	0.955
E	8	54	40	36	133	95	76	228	398.994	1,424.527	0.0190	26.991	75,183.394	4,176.855	0.511
E	9	98	30	71	326	246	101	572	583.839	3,808.497	0.0190	73.280	197,934.561	10,996.364	1.410
E	10	52	35	34	114	81	69	195	352.499	1,191.162	0.0240	28.571	49,660.956	2,758.942	0.685
F	1	41	30	41	112	89	71	201	366.866	1,236.099	0.0310	37.963	40,248.595	2,236.033	1.166
F	2	45	38	38	106	59	76	165	398.842	1,030.908	0.0340	34.946	30,411.787	1,689.544	1.185
F	3	73	35	63	203	140	98	343	556.094	2,268.845	0.0260	58.342	88,232.880	4,901.827	1.500
F	4	70	36	49	149	124	85	273	461.238	1,749.764	0.0220	38.413	79,703.523	4,427.974	0.843
F	5	47	30	23	101	68	53	169	251.249	968.019	0.0230	21.828	42,929.519	2,384.973	0.492
F	6	40	29	34	97	56	63	153	313.855	914.524	0.0420	38.293	21,840.981	1,213.388	1.603
F	7	39	37	33	84	56	70	140	359.215	858.100	0.0320	27.333	26,939.128	1,496.618	0.871
F	8	40	34	32	97	61	66	158	332.974	955.014	0.0310	29.470	30,948.431	1,719.357	0.909
F	9	94	28	72	335	231	100	566	578.841	3,760.423	0.0220	83.720	168,905.649	9,383.647	1.864
F	10	53	35	37	104	80	72	184	372.275	1,135.266	0.0260	30.003	42,956.018	2,386.445	0.793

Tabla 5.2 Datos Generales GC ordenado por programador cont...

Grupo Experimental															
Progr	#Prog	LOC	n1	n2	N1	N2	n	N	N^	Volumen	Nivel Prog	Cont. Int	Esfuerzo	Tpo. Est. Prog.	Nivel Leng
C	7	24	25	23	52	38	48	90	220.138	502.647	0.0480	24.339	10,380.746	576.708	1.179
D	7	24	24	26	52	40	50	92	232.251	519.235	0.054	28.125	9,585.873	532.548	1.523
F	7	30	25	29	72	50	54	122	256.978	702.096	0.0460	32.577	15,131.385	840.633	1.512
D	2	28	31	30	77	49	61	126	300.787	747.273	0.039	29.517	18,918.459	1,051.026	1.166
C	5	30	28	24	84	54	52	138	244.645	786.661	0.0320	24.973	24,779.811	1,376.656	0.793
D	5	33	29	23	86	53	52	139	244.923	792.361	0.030	23.714	26,475.197	1,470.844	0.710
E	7	31	26	32	81	59	58	140	282.211	820.117	0.0420	34.216	19,657.187	1,092.066	1.428
C	2	33	31	31	86	64	62	150	307.160	893.129	0.0310	27.910	28,580.142	1,587.786	0.872
E	1	28	23	31	90	69	54	159	257.622	915.027	0.0390	35.748	23,421.742	1,301.208	1.397
B	7	38	30	34	97	63	64	160	320.180	960.000	0.0360	34.540	26,682.353	1,482.353	1.243
D	10	38	30	32	95	65	62	160	307.207	952.671	0.033	31.267	29,026.707	1,612.595	1.026
A	7	39	31	32	99	66	63	165	313.58	986.251	0.0310	30.85	31,529.218	1,751.623	0.965
A	5	37	33	30	98	68	63	166	313.672	992.228	0.0270	26.53	37,109.345	2,061.630	0.709
D	1	32	25	35	95	71	50	166	295.621	980.544	0.039	38.669	24,863.790	1,381.322	1.525
B	5	36	31	30	98	69	61	167	300.787	990.433	0.0280	27.782	35,308.941	1,961.608	0.779
B	1	29	24	35	99	72	59	171	289.564	1,005.932	0.0410	40.750	24,832.149	1,379.564	1.651
E	5	39	30	30	96	75	60	171	294.413	1,010.078	0.0270	26.935	37,877.936	2,104.330	0.718
B	8	35	36	27	104	70	63	174	314.499	1,040.047	0.0210	22.287	48,535.513	2,696.417	0.478
A	1	31	26	36	101	74	62	175	308.329	1,041.948	0.0370	38.993	27,844.137	1,546.897	1.459
F	2	38	30	38	103	73	68	176	346.628	1,071.393	0.0350	37.181	30,873.048	1,715.169	1.290
B	10	40	33	34	102	76	67	178	339.439	1,079.764	0.0270	29.276	39,824.232	2,212.457	0.794
F	5	40	30	28	102	76	58	178	281.813	1,042.721	0.0250	25.611	42,453.625	2,358.535	0.629
E	2	35	29	37	107	73	66	180	333.631	1,087.991	0.0350	38.031	31,125.362	1,729.187	1.329
E	10	44	32	28	103	79	60	182	294.606	1,075.054	0.0220	23.814	48,531.013	2,696.167	0.528
A	8	35	36	25	110	73	61	183	302.214	1,085.325	0.0190	20.649	57,044.678	3,169.149	0.393
B	6	38	33	35	112	74	68	186	345.990	1,132.268	0.0290	32.456	39,499.981	2,194.433	0.930
D	6	38	27	34	124	70	61	194	301.356	1,150.563	0.036	41.396	31,978.885	1,776.605	1.489
C	10	44	31	31	113	82	62	195	307.160	1,161.068	0.0240	28.319	47,603.800	2,644.656	0.691
A	6	39	34	34	120	77	68	197	345.947	1,199.230	0.0260	31.149	46,170.362	2,565.020	0.809
F	10	43	30	35	115	83	65	198	326.732	1,192.429	0.0280	33.522	42,416.397	2,356.466	0.942
F	1	34	27	36	113	86	63	199	314.499	1,189.479	0.0310	36.883	38,360.688	2,131.149	1.144
F	6	39	28	36	117	82	64	199	320.723	1,194.000	0.0310	37.443	38,075.333	2,115.296	1.174
A	10	45	37	34	117	83	71	200	365.724	1,229.949	0.0220	27.234	55,546.686	3,085.927	0.603
B	2	43	37	40	122	85	77	207	405.627	1,297.225	0.0250	32.998	50,997.150	2,833.175	0.839
C	8	38	30	32	125	83	62	208	307.207	1,238.473	0.0260	31.832	48,184.334	2,676.907	0.818
C	1	40	23	34	116	95	57	211	277.016	1,230.740	0.0310	38.302	39,546.565	2,197.031	1.192
C	6	41	27	38	133	84	65	217	327.803	1,306.854	0.0340	43.792	38,999.269	2,166.626	1.467
F	8	43	32	36	123	94	68	217	346.117	1,320.979	0.0240	31.619	55,187.585	3,065.977	0.757
D	4	46	32	41	124	94	73	218	379.660	1,349.382	0.027	36.785	49,499.272	2,749.960	1.003
A	2	49	39	44	129	94	83	223	446.346	1,421.634	0.0240	34.125	59,223.971	3,290.221	0.819
E	8	44	30	41	141	91	71	232	366.866	1,426.741	0.0300	42.855	47,500.047	2,638.891	1.287
B	4	48	35	39	133	102	74	235	385.656	1,459.222	0.0220	31.882	66,787.447	3,710.414	0.697

Tabla 5.3 Datos Generales GE ordenado por N

A	4	51	38	39	135	103	77	238	405.552	1,491.495	0.0200	29.723	74,842.464	4,157.915	0.592
E	4	55	32	40	138	111	72	249	372.877	1,536.311	0.0230	34.602	68,212.223	3,789.568	0.779
E	6	49	25	42	162	97	67	259	342.574	1,571.117	0.0350	54.422	45,356.654	2,519.814	1.885
C	4	58	33	44	167	131	77	298	406.680	1,867.502	0.0200	38.015	91,741.055	5,096.725	0.774
F	4	63	31	38	159	142	69	301	353.001	1,838.666	0.0170	31.744	106,497.462	5,916.526	0.548
D	8	57	31	43	193	116	74	309	386.909	1,918.721	0.024	45.887	80,229.314	4,457.184	1.097
D	3	60	31	66	217	139	97	356	552.510	2,349.569	0.031	71.976	76,699.187	4,261.066	2.205
C	3	67	35	63	226	154	98	380	556.094	2,513.590	0.0230	58.759	107,525.783	5,973.655	1.374
A	3	71	37	67	227	162	104	389	599.178	2,606.471	0.0220	58.269	116,590.951	6,477.275	1.303
B	3	75	38	65	231	164	103	395	590.875	2,641.168	0.0210	55.095	126,613.517	7,034.084	1.149
E	3	71	30	67	239	163	97	402	553.635	2,653.165	0.02700	72.704	96,820.721	5,378.929	1.992
F	3	73	35	66	227	179	101	406	578.455	2,703.234	0.0210	56.956	128,301.213	7,127.845	1.200
F	9	67	22	67	258	194	89	452	504.535	2,927.032	0.0310	91.898	93,228.138	5,179.341	2.885
D	9	82	25	61	321	218	86	539	477.871	3,463.757	0.022	77.537	154,733.394	8,596.300	1.736
A	9	96	27	70	337	237	97	574	557.432	3,788.350	0.0220	82.883	173,154.653	9,619.703	1.813
E	9	91	23	69	354	245	92	599	525.530	3,907.614	0.0240	95.697	159,560.889	8,864.494	2.344
C	9	105	22	83	341	271	105	612	627.236	4,109.118	0.0280	114.410	147,581.705	8,198.984	3.186
B	9	106	26	70	363	272	96	635	551.261	4,181.451	0.0200	82.778	211,223.020	11,734.612	1.639

Tabla 5.3 Datos Generales GE ordenado por N cont...

GRUPO DE CONTROL															
Progr	#Prog	LOC	n1	n2	N1	N2	n	N	N^	Volumen	Nivel Prog	Cont. Int.	Esfuerzo	Tpo. Est. Prog.	Nivel Leng
C	7	24	25	23	52	38	48	90	220.138	502.647	0.0480	24.339	10,380.746	576.708	1.179
E	7	27	30	26	62	44	56	106	269.418	615.580	0.0390	24.250	15,626.252	868.125	0.955
D	2	28	31	29	79	51	60	130	294.462	767.896	0.0370	28.171	20,931.780	1,162.877	1.033
D	7	43	21	22	63	68	43	131	190.346	710.841	0.0310	21.903	23,070.011	1,281.011	0.675
F	7	39	37	33	84	56	70	140	359.215	858.100	0.0320	27.333	26,939.128	1,496.618	0.871
C	5	31	28	24	85	56	52	141	244.645	803.762	0.0310	24.605	26,256.225	1,458.679	0.753
A	7	39	33	36	88	58	69	146	352.582	891.845	0.0380	33.549	23,708.202	1,317.122	1.262
E	2	43	38	31	94	57	69	151	353.001	922.387	0.0290	26.403	32,224.043	1,790.225	0.756
C	2	33	31	31	87	65	62	152	307.160	905.038	0.0310	27.847	29,413.730	1,634.096	0.857
F	6	40	29	34	97	56	63	153	313.855	914.524	0.0420	38.293	21,840.981	1,213.388	1.603
F	8	40	34	32	97	61	66	158	332.974	955.014	0.0310	29.470	30,948.431	1,719.357	0.909
B	7	38	30	29	97	63	59	160	288.088	941.223	0.0310	28.884	30,670.884	1,703.938	0.886
A	5	41	29	27	99	64	56	163	269.263	946.599	0.0290	27.541	32,534.953	1,807.497	0.801
F	2	45	38	38	106	59	76	165	398.842	1,030.908	0.0340	34.946	30,411.787	1,689.544	1.185
B	5	36	31	30	98	69	61	167	300.787	990.433	0.0280	27.782	35,308.941	1,961.608	0.779
F	5	47	30	23	101	68	53	169	251.249	968.019	0.0230	21.828	42,929.519	2,384.973	0.492
B	1	29	24	35	99	72	59	171	289.564	1,005.932	0.0410	40.750	24,832.149	1,379.564	1.651
B	8	35	36	26	104	70	62	174	308.329	1,036.030	0.0210	21.378	50,207.615	2,789.312	0.441
B	10	40	34	32	103	77	66	180	332.974	1,087.991	0.0240	26.597	44,505.629	2,472.535	0.650
D	5	46	25	23	104	76	48	180	220.138	1,005.293	0.0240	24.339	41,522.982	2,306.832	0.589
F	10	53	35	37	104	80	72	184	372.275	1,135.266	0.0260	30.003	42,956.018	2,386.445	0.793
A	2	48	38	43	117	68	81	185	432.751	1,172.872	0.0330	39.035	35,240.720	1,957.818	1.299
B	6	38	33	34	112	74	67	186	339.439	1,128.293	0.0280	31.418	40,518.978	2,251.054	0.875
E	5	46	32	30	109	78	62	187	307.207	1,113.435	0.0240	26.765	46,318.884	2,573.271	0.643
C	10	44	31	31	113	82	62	195	307.160	1,161.068	0.0240	28.319	47,603.800	2,644.656	0.691
E	10	52	35	34	114	81	69	195	352.499	1,191.162	0.0240	28.571	49,660.956	2,758.942	0.685
A	6	45	30	42	124	72	72	196	373.684	1,209.305	0.0390	47.029	31,096.422	1,727.579	1.829
F	1	41	30	41	112	89	71	201	366.866	1,236.099	0.0310	37.963	40,248.595	2,236.033	1.166
A	8	46	35	42	125	79	77	204	406.002	1,278.424	0.0300	38.838	42,081.472	2,337.860	1.180
E	1	38	30	34	115	91	64	206	320.180	1,236.000	0.0250	30.787	49,621.765	2,756.765	0.767
C	8	38	30	32	125	83	62	208	307.207	1,238.473	0.0260	31.832	48,184.334	2,676.907	0.818
C	1	40	23	34	116	95	57	211	277.016	1,230.740	0.0310	38.302	39,546.565	2,197.031	1.192
A	10	59	35	42	126	91	77	217	406.002	1,359.893	0.0260	35.865	51,562.597	2,864.589	0.946
C	6	41	27	38	133	84	65	217	327.803	1,306.854	0.0340	43.792	38,999.269	2,166.626	1.467
B	2	47	36	41	128	92	77	220	405.777	1,378.693	0.0250	34.134	55,685.748	3,093.653	0.845
A	1	45	30	46	126	100	76	226	401.291	1,412.032	0.0310	43.302	46,044.509	2,558.028	1.328
E	8	54	40	36	133	95	76	228	398.994	1,424.527	0.0190	26.991	75,183.394	4,176.855	0.511
B	4	47	35	38	132	100	73	232	378.946	1,436.039	0.0220	31.183	66,133.389	3,674.077	0.677
E	6	56	35	41	139	97	76	236	399.185	1,474.511	0.0240	35.614	61,048.347	3,391.575	0.860
D	10	58	36	39	139	103	75	242	392.248	1,507.374	0.0210	31.709	71,658.247	3,981.014	0.667
D	1	46	31	37	139	107	68	246	346.330	1,497.516	0.0220	33.409	67,125.137	3,729.174	0.745
F	4	70	36	49	149	124	85	273	461.238	1,749.764	0.0220	38.413	79,703.523	4,427.974	0.843

Tabla 5.4 Datos Generales GC ordenado por N

D	8	55	38	51	172	118	89	290	488.715	1,877.963	0.0230	42.719	82,556.713	4,586.484	0.972
E	4	73	38	45	159	133	83	292	446.555	1,861.512	0.0180	33.149	104,534.213	5,807.456	0.590
C	4	58	33	44	167	131	77	298	406.680	1,867.502	0.0200	38.015	91,741.055	5,096.725	0.774
A	4	76	36	54	167	139	90	306	496.881	1,986.507	0.0220	42.874	92,041.493	5,113.416	0.925
D	6	63	33	58	200	129	91	329	506.228	2,141.064	0.0270	58.342	78,573.373	4,365.187	1.590
F	3	73	35	63	203	140	98	343	556.094	2,268.845	0.0260	58.342	88,232.880	4,901.827	1.500
D	4	73	39	49	196	156	88	352	481.251	2,273.720	0.0160	36.625	141,156.245	7,842.014	0.590
A	3	82	39	65	218	152	104	370	597.585	2,479.163	0.0220	54.368	113,049.819	6,280.545	1.192
C	3	67	35	63	226	154	98	380	556.094	2,513.590	0.0230	58.759	107,525.783	5,973.655	1.374
B	3	75	38	63	231	164	101	395	575.990	2,629.994	0.0200	53.174	130,080.315	7,226.684	1.075
E	3	91	40	66	229	175	106	404	611.807	2,718.080	0.0190	51.255	144,140.599	8,007.811	0.967
D	3	83	36	78	246	189	116	435	689.683	2,983.222	0.0220	64.798	137,342.939	7,630.163	1.407
D	9	80	21	59	320	227	80	547	493.315	3,458.095	0.0250	85.600	139,701.163	7,761.176	2.119
F	9	94	28	72	335	231	100	566	578.841	3,760.423	0.0220	83.720	168,905.649	9,383.647	1.864
E	9	98	30	71	326	246	101	572	583.839	3,808.497	0.0190	73.280	197,934.561	10,996.364	1.410
C	9	105	22	83	341	271	105	612	627.236	4,109.118	0.0280	114.410	147,581.705	8,198.984	3.186
B	9	106	26	70	363	272	96	635	551.261	4,181.451	0.0200	82.778	211,223.020	11,734.612	1.639
A	9	108	28	74	367	275	102	642	594.105	4,283.697	0.0190	82.336	222,868.023	12,381.557	1.583

5.4 ESTADÍSTICAS DESCRIPTIVAS

Mediante el análisis de los datos obtenidos, se lograron resultados que permiten la aceptación de 4 de las 6 hipótesis propuestas con diferentes grados de confiabilidad. Es importante destacar que de las dos hipótesis no aceptadas, si bien no hay suficiente prueba estadística que permita su aceptación, sí hay tendencias a favor de cada una de ellas.

Antes de hacer el análisis de cada una de las hipótesis se dará una descripción estadística general para comparar el *GE.* y el *GC.*

Aunque no es parte de la investigación, es interesante ver la correlación entre ***N*** y ***N^*** para cada uno de los grupos; la correlación que Martínez [1994] encontró para FOXPRO2 fue de 0.96215. En este caso en particular se encontró.

Grupo	Corr. Pearson
Gpo. Experimental	0.8858
Gpo. Control	0.8519

Tabla 5.5: Coeficiente de Correlación Pearson entre *N* y *N^* para *GE.* y *GC.*

Por otro lado, las diferencias pueden notarse también en las ***LOC, n1, n2, N1, N2, n, N y N^***.

Gpo. Exp.	LOC	n1	n2	N1	N2
SUMA	2,856	1,801	2,440	8,661	6,176
PROMEDIO	47.60	30.01	40.66	144.35	102.93
DESV. EST.	19.38	4.40	14.71	76.12	56.03
VARIANZA	375.76	19.40	216.59	5,794.90	3,139.55

Tabla 5.6 Resultados de LOC, n1, n2, N1, N2 para el Grupo Experimental.

Gpo.Control	LOC	n1	n2	N1	N2
SUMA	3,256	1,934	2,553	8,995	6,495
PROMEDIO	54.26	32.23	42.55	149.91	108.25
DES.EST.	21.14	4.90	15.56	77.92	59.72
VARIANZA	447.08	24.01	242.35	6,072.41	3,567.41

Tabla 5.7 Resultados de LOC, n1, n2, N1, N2 para el Grupo Control.

En las tablas 5.6 y 5.7 podemos ver, por ejemplo, que el *GE.* tiende a usar menos líneas de código que el *GC*, la desviación estándar es menor, lo cual habla de mayor estabilidad. Así, para cada uno de los rubros, el *GE* tiende a usar menos operandos y menos operadores.

Gpo. Exp.	n	N	N^
SUMA	4,241	14,837	22,115.53
PROMEDIO	70.68	247.28	368.59
DESV. EST.	15.09	131.69	105.41
VARIANZA	227.94	17,343.79	11,111.92

Tabla 5.8 : Resultados n, N y N^ para el grupo de Experimental.

Gpo.Cont.	n	N	N^
SUMA	4,487	15,490	23,819.29
PROMEDIO	74.78	258.16	396.98
DES.EST.	16.87	137.21	117.50
VARIANZA	284.85	18,827.02	13,808.04

Tabla 5.9 : Resultados n, N y N^ para el grupo de Control.

De esta manera, en las tablas 5.8 y 5.9 podemos observar la misma tendencia; la razón de este comportamiento es que entre la N y la $N^{\wedge}$ del *GC* hay más desviación que en el *GE*, confirmando por consecuencia, que los programas del *GC* tienen más impurezas que los del *GE* tal como Halstead lo hipotetizó [Harrinson W, 1998]. Tales impurezas son:

- Operadores de Cancelación (canceling of operators)
- Operandos Ambiguos
- Operandos Sinónimos
- Subexpresiones Comunes
- Reemplazos innecesarios
- Expresiones no Factorizadas

Con esto podemos decir que el *GE* tuvo más disciplina al usar tanto operadores como operandos, tal disciplina está directamente relacionada con el curso de métricas de software que se les impartió.

5.5 ANÁLISIS DE DATOS DE LA PRIMERA HIPÓTESIS DE INVESTIGACIÓN

Las hipotesis planteadas son:

Ho: La media del volumen del programa de los programas hechos por el *GE.* es igual a la media del volumen del programa de los programas hechos por el *GC.*

Ha: La media del volumen del programa de los programas hechos por el *GE.* es menor que la media del volumen del programa de los programas hechos por el *GC.*

Prueba z para medias de dos muestras		
α=0.20	Gpo. Exp.	Gpo.Cont.
	Volumen	*Volumen*
Media	1,544.16	1,632.35
Varianza (conocida)	820,011.07	902,541.70
Observaciones	60	60
Diferencia hipotética de las medias	0	
z	-0.52048086	
P(Z<=z) una cola	0.30136422	
Valor crítico de z (una cola)	0.84162139	
P(Z<=z) dos colas	0.15068211	
Valor crítico de z (dos colas)	1.28155079	

Tabla 5.10: Prueba Z para el Volumen del Programa

Dado que tenemos una alternativa de cola inferior, el área de rechazo está cuando $Z < -Z\alpha$, entonces tenemos que: -0.52048086< -0.84162139. Como la condición anterior es falsa, entonces se acepta la Ho y se rechaza la Ha. Sin embargo, el valor de Z tiende hacia el área de rechazo, es decir hay una tendencia en favor de la Ha; si el valor de Z fuera positivo, no habría tendencia en favor de la Ha.

Conclusión #1: Se encontró que la tendencia a favor de la Ha, con un nivel de confianza del 80%, es del 61.83%. Sin embargo, no hay suficiente prueba estadística que permita aceptar que la media del volumen del programa de los programas hechos por el grupo experimental sea menor que la media del volumen del programa de los programas hechos por el grupo de control.

Las distribuciones de Frecuencia para el Volumen del Programa para cada Grupo son:

Gpo. Exp.	Volumen		
Clase	*Frecuencia*	*Porcentaje*	*% acumulado*
500-1000	15	25.00	25.00%
1001-1500	28	46.67	71.67%
1501-2000	5	8.33	80.00%
2001-2500	1	1.67	81.67%
2501-3000	6	10.00	91.67%
3001-3500	1	1.67	93.33%
3501-4000	2	3.33	96.67%
4001-4500	2	3.33	100.00%

Tabla 5.11: Distribución de Frecuencia para el Volumen del Programa del Grupo Experimental

Gpo. Control	Volumen		
Clase	*Frecuencia*	*Porcentaje*	*% acumulado*
500-1000	15	25.00	25.00%
1001-1500	25	41.67	66.67%
1501-2000	6	10.00	76.67%
2001-2500	4	6.67	83.33%
2501-3000	4	6.67	90.00%
3001-3500	1	1.67	91.67%
3501-4000	2	3.33	95.00%
4001-4500	3	5.00	100.00%

Tabla 5.12: Distribución de Frecuencia para el Volumen del Programa del Grupo de Control.

5.6 ANÁLISIS DE DATOS DE LA SEGUNDA HIPÓTESIS DE INVESTIGACIÓN

Las hipotesis planteadas son:

Ho: La media del nivel del programa de los programas hechos por el *GE.* es igual la media del nivel del programa de los programas hechos por el *GC.*

Ha: La media del nivel del programa de los programas hechos por el *GE.* es mayor que la media del nivel del programa de los programas hechos por el *GC.*

Prueba z para medias de dos muestras		
α=0.10	Gpo. Exp.	Gpo.Cont.
	Nivel Prog	*Nivel Prog*
Media	0.02908333	0.02703333
Varianza (conocida)	5.8993E-05	4.488E-05
Observaciones	60	60
Diferencia hipotética de las medias	0	
z	1.55803762	
P(Z<=z)una cola	0.05961218	
Valor crítico de z (una cola)	1.28155079	
P(Z<=z) dos colas	0.02980609	
Valor crítico de z (dos colas)	1.644853	

Tabla 5.13: Prueba Z para el Nivel del Programa

Dado que tenemos una alternativa de cola superior, el área de rechazo está cuando $Z > Z\alpha$, entonces tenemos que: 1.55803762> 1.28155079. Como la condición anterior es verdadera, entonces se rechaza la Ho y se acepta Ha.

Conclusión #2: Hay suficiente prueba estadística, con una confiabilidad del 90%, que permite aceptar que la media del nivel del programa de los programas hechos por el grupo experimental es mayor que la media del nivel del programa de los programas hechos por el grupo de control.

Las distribuciones de Frecuencia para el Nivel del Programa para cada Grupo son:

Gpo. Exp.	Nivel Prog.		
Clase	*Frecuencia*	*Porcentaje*	*% acumulado*
0.000-0.010	0	0	.00%
0.011-0.020	5	8.33	8.33%
0.021-0.030	31	51.67	60.00%
0.031-0.040	19	31.67	91.67%
0.041-0.050	4	6.67	98.33%
0.051-0.060	1	1.67	100.00%

Tabla 5.14: Distribución de Frecuencia para el Nivel del Programa del Grupo de Experimental.

Gpo. Control	Nivel Prog.		
Clase	*Frecuencia*	*Porcentaje*	*% acumulado*
0.00-0.010	0	0	.00%
0.011-0.020	9	15.00	15.00%
0.021-0.030	32	53.33	68.33%
0.031-0.040	16	26.67	95.00%
0.041-0.050	3	5.00	100.00%
0.051-0.060	0	0.00	100.00%

Tabla 5.15: Distribución de Frecuencia para el Nivel del Programa del Grupo de Control.

5.7 ANÁLISIS DE DATOS DE LA TERCERA HIPÓTESIS DE INVESTIGACIÓN

Las hipótesis planteadas son:

Ho: La media del contenido de inteligencia de los programas hechos por el *GE.* es igual la media del contenido de inteligencia de los programas hechos por el *GC.*

Ha: La media del contenido de inteligencia de los programas hechos por el *GE.* es mayor que la media contenido de inteligencia de los programas hechos por el *GC.*

Prueba z para medias de dos muestras		
α=0.20	Gpo. Exp.	Gpo. Control
	Cont. Int	*Cont. Int.*
Media	41.6705167	40.6337667
Varianza (conocida)	407.13717	358.369707
Observaciones	60	60
Diferencia hipotética de las medias	0	
z	0.29025193	
P(Z<=z) una cola	0.38581182	
Valor crítico de z (una cola)	0.84162139	
P(Z<=z) dos colas	0.19290591	
Valor crítico de z (dos colas)	1.28155079	

Tabla 5.16: Prueba Z para el Contenido de Inteligencia

Dado que tenemos una alternativa de cola superior, el área de rechazo está cuando $Z > Z\alpha$, entonces tenemos que: 0.29025193> 0.84162139. Como la condición anterior es falsa, entonces se acepta la Ho y se rechaza Ha. Sin embargo, el valor de Z tiende hacia el área de rechazo, es decir hay una tendencia en favor de la Ha; si el valor de Z fuera negativa, no habría tendencia en favor de la Ha.

Conclusión #3: Se encontró que la tendencia a favor de la Ha, con un nivel de confianza del 80%, es del 34.48%. Sin embargo, no hay suficiente prueba estadística que permita aceptar que la media del contenido de inteligencia de los programas hechos por el grupo de experimental sea mayor que la media del contenido de inteligencia de los programas hechos por el grupo de control.

Las distribuciones de Frecuencia para el Contenido de Inteligencia para cada Grupo son:

Gpo. Exp.	Cont. Inteligencia		
Clase	*Frecuencia*	*Porcentaje*	*% acumulado*
0-10	0	0.00	.00%
11-20	0	0.00	.00%
21-30	17	28.33	28.33%
31-40	25	41.67	70.00%
41-50	5	8.33	78.33%
51-60	5	8.33	86.67%
61-70	0	0.00	86.67%
71-80	3	5.00	91.67%
81-90	2	3.33	95.00%
91-100	2	3.33	98.33%
101-110	0	0.00	98.33%
111-120	1	1.67	100.00%

Tabla 5.17: Distribución de Frecuencia para el Contenido de Inteligencia del Grupo de Experimental.

Gpo. Control	Cont. Inteligencia		
Clase	*Frecuencia*	*Porcentaje*	*% acumulado*
0-10	0	0.00	.00%
11-20	0	0.00	.00%
21-30	20	33.33	33.33%
31-40	21	35.00	68.33%
41-50	6	10.00	78.33%
51-60	6	10.00	88.33%
61-70	1	1.67	90.00%
71-80	1	1.67	91.67%
81-90	4	6.67	98.33%
91-100	0	0.00	98.33%
101-110	0	0.00	98.33%
111-120	1	1.67	100.00%

Tabla 5.18: Distribución de Frecuencia para el Contenido de Inteligencia del Grupo de Control.

5.8 ANÁLISIS DE DATOS DE LA CUARTA HIPÓTESIS DE INVESTIGACIÓN

Las hipotesis planteadas son:

Ho: La media del esfuerzo de programación de los programas hechos por el GE. es igual la media del esfuerzo programación de los programas hechos por el GC.

Ha: La media del esfuerzo de programación de los programas hechos por el GE. es menor la media del esfuerzo programación de los programas hechos por el GC.

Prueba z para medias de dos muestras		
α=0.20	Gpo.Exp.	Gpo.Control
	Esfuerzo	*Esfuerzo*
Media	60,081.3784	68,661.2701
Varianza (conocida)	1,924,678,967	2,555,900,284
Observaciones	60	60
Diferencia hipotética de las medias	0	
z	-0.99286534	
P(Z<=z) una cola	0.1603878	
Valor crítico de z (una cola)	0.84162139	
P(Z<=z) dos colas	0.0801939	
Valor crítico de z (dos colas)	1.28155079	

Tabla 5.19: Prueba Z para el Esfuerzo de Programación

Dado que tenemos una alternativa de cola inferior, el área de rechazo está cuando $Z < -Z\alpha$, entonces tenemos que: -0.99286534 < -0.84162139. Como la condición anterior es verdadera, entonces se rechaza Ho y se acepta la Ha.

Conclusión #4: Hay suficiente prueba estadística, con una confiabilidad del 80%, que permite aceptar que la media del esfuerzo programación de los programas hechos por el grupo experimental es menor que la media a la media del esfuerzo de programación de los programas hechos por el grupo de control.

Las distribuciones de frecuencia del Esfuerzo de Programación para cada Grupo son:

Gpo. Exp.	Esfuerzo		
Clase	*Frecuencia*	*Porcentaje*	*% acumulado*
0 - 20,000	5	8.33	8.33%
20,001 - 40,000	22	36.67	45.00%
40,001 - 60,000	15	25.00	70.00%
60,001 - 80,000	4	6.67	76.67%
80,001 - 100,000	4	6.67	83.33%
100,001 - 120,000	3	5.00	88.33%
120,001 - 140,000	2	3.33	91.67%
140,001 - 160,000	3	5.00	96.67%
160,001 - 180,000	1	1.67	98.33%
180,001 - 200,000	0	-	98.33%
200,001 - 220,000	1	1.67	100.00%
220,001 - 240,000	0	-	100.00%

Tabla 5.20: Distribución de Frecuencia para el Esfuerzo de Programación del Grupo de Experimental.

Gpo. Control	Esfuerzo		
Clase	*Frecuencia*	*Porcentaje*	*% acumulado*
0 - 20,000	2	3.33	3.33%
20,001 - 40,000	18	30.00	33.33%
40,001 - 60,000	16	26.67	60.00%
60,001 - 80,000	7	11.67	71.67%
80,001 - 100,000	4	6.67	78.33%
100,001 - 120,000	3	5.00	83.33%
120,001 - 140,000	3	5.00	88.33%
140,001 - 160,000	3	5.00	93.33%
160,001 - 180,000	1	1.67	95.00%
180,001 - 200,000	1	1.67	96.67%
200,001 - 220,000	1	1.67	98.33%
220-001 - 240,000	1	1.67	100.00%

Tabla 5.21: Distribución de Frecuencia para el Esfuerzo de Programación del Grupo de Control.

5.9 ANÁLISIS DE DATOS DE LA QUINTA HIPÓTESIS DE INVESTIGACIÓN

Las hipotesis planteadas son:

Ho: La media del tiempo estimado de programación de los programas hechos por el *GE.* es igual a la media de tiempo estimado programación de los programas hechos por el *GC.*

Ha: La media del tiempo estimado de programación de los programas hechos por el *GE.* es menor a la media del tiempo estimado programación de los programas hechos por el *GC.*

Prueba z para medias de dos muestras		
α=0.20	Gpo. Exp.	Gpo. Control
	Tiempo Est. Prog.	*Tiempo Est. Prog.*
Media	3,337.8542	3,814.504033
Varianza (conocida)	5,940,367.67	7,888,637.43
Observaciones	60	60
Diferencia hipotética de las medias	0	
z	-0.992840769	
P(Z<=z) una cola	0.160393782	
Valor crítico de z (una cola)	0.841621386	
P(Z<=z) dos colas	0.080196891	
Valor crítico de z (dos colas)	1.281550794	

Tabla 5.22: Prueba Z para el tiempo Estimado de Programación

Dado que tenemos una alternativa de cola inferior, el área de rechazo está cuando $Z < -Z\alpha$, entonces tenemos que: -0.99286534 < -0.841621386. Como la condición anterior es verdadera, entonces se acepta la Ha y se rechaza Ho.

Conclusión #5: Hay suficiente prueba estadística, con una confiabilidad del 80%, que permite aceptar que la media del tiempo estimado programación de los programas hechos por el grupo experimental es menor que la media del tiempo estimado de programación de los programas hechos por el grupo de control.

Las distribuciones de Frecuencia del Tiempo Estimado de Programación para cada Grupo son:

Gpo. Exp.	Tpo. Est. Prog.		
Clase	*Frecuencia*	*Porcentaje*	*% acumulado*
0 - 1,000	3	5.00	5.00%
1,001 - 2,000	16	26.67	31.67%
2,001 - 3,000	19	31.67	63.33%
3,001 - 4,000	6	10.00	73.33%
4,001 - 5,000	3	5.00	78.33%
5,001 - 6,000	5	8.33	86.67%
6,001 - 7,000	1	1.67	88.33%
7,001 - 8,000	2	3.33	91.67%
8,001 - 9,000	3	5.00	96.67%
9,001 - 10,000	1	1.67	98.33%
10,001 - 11,000	0	-	98.33%
11,001 - 12,000	1	1.67	100.00%
12,001 - 13,000	0	-	100.00%

Tabla 5.23: Distribución de Frecuencia para el Tiempo Estimado de Programación del Grupo de Experimental.

Gpo. Control	Tpo.Est.Prog.		
Clase	*Frecuencia*	*Porcentaje*	*% acumulado*
0 - 1,000	2	3.33	3.33%
1,001 - 2,000	16	26.67	30.00%
2,001 - 3,000	17	28.33	58.33%
3,001 - 4,000	5	8.33	66.67%
4,001 - 5,000	5	8.33	75.00%
5,001 - 6,000	4	6.67	81.67%
6,001 - 7,000	1	1.67	83.33%
7,001 - 8,000	4	6.67	90.00%
8,001 - 9,000	2	3.33	93.33%
9,001 - 10,000	1	1.67	95.00%
10,001 - 11,000	1	1.67	96.67%
11,001 - 12,000	1	1.67	98.33%
12,001 - 13,000	1	1.67	100.00%

Tabla 5.24: Distribución de Frecuencia para el Tiempo Estimado de Programación del Grupo de Control.

5.10 ANÁLISIS DE DATOS DE LA SEXTA HIPÓTESIS DE INVESTIGACIÓN

Las hipótesis planteadas son:

Ho: La media del nivel del lenguaje de los programas hechos por el *GE.* es igual la media del nivel del lenguaje de los programas hechos por el *GC.*

Ha: La media del nivel del lenguaje de los programas hechos por el *GE.* es mayor que la media del nivel del lenguaje de los programas hechos por el *GC.*

Prueba z para medias de dos muestras		
α=0.10		
	Nivel Leng	*Nivel Leng*
Media	1.18763333	1.06518333
Varianza (conocida)	0.3101298	0.22368863
Observaciones	60	60
Diferencia hipotética de las medias	0	
z	1.29818814	
P(Z<=z) una cola	0.09711141	
Valor crítico de z (una cola)	1.28155079	
P(Z<=z) dos colas	0.04855571	
Valor crítico de z (dos colas)	1.644853	

Tabla 5.25: Prueba Z para el Nivel del Lenguaje

Dado que tenemos una alternativa de cola superior, el área de rechazo está cuando $Z > Z\alpha$, entonces tenemos que: 1.29818814 > 1.28155079. Como la condición anterior es verdadera, entonces se acepta la Ha y se rechaza Ho.

Conclusión #6: Hay suficiente prueba estadística, con una confiabilidad del 90%, que permite aceptar que la media del nivel del lenguaje de los programas hechos por el grupo experimental es mayor que la media del nivel del lenguaje de los programas hechos por el grupo de control.

Las distribuciones de Frecuencia del Nivel del Leguaje para cada Grupo son:

Gpo. Exp.	Nivel Leng.		
Clase	*Frecuencia*	*Porcentaje*	*% acumulado*
0-0.5	2	3.33	3.33%
0.51-1.00	24	40.00	43.33%
1.01-1.50	21	35.00	78.33%
1.51-2.00	9	15.00	93.33%
2.01-2.50	2	3.33	96.67%
2.51-3.00	1	1.67	98.33%
3.01-3.50	1	1.67	100.00%

Tabla 5.26: Distribución de Frecuencia para el Nivel del Lenguaje del Grupo de Experimental.

Gpo.Control	Nivel Leng.		
Clase	*Frecuencia*	*Porcentaje*	*% acumulado*
0-0.5	2	3.33	3.33%
0.51-1.00	33	55.00	58.33%
1.01-1.50	16	26.67	85.00%
1.51-2.00	7	11.67	96.67%
2.01-2.50	1	1.67	98.33%
2.51-3.00	0	-	98.33%
3.01-3.50	1	1.67	100.00%

Tabla 5.27: Distribución de Frecuencia para el Nivel del Lenguaje del Grupo de Experimental.

5.11 RESUMEN DEL CAPÍTULO

En este capítulo se presentó el análisis de los datos que se recolectaron. El resumen es el siguiente:

1) La diferencia en la correlación de N y $N^{\wedge}$ entre el *GE* y *GC* es del 3.39% donde la correlación más alta fue para el *GE*.
2) Se puede ver que los programas del *GC* tienen más impurezas que los del *GE*. y en general, en todos los rubros, los programas del *GE* mostraron una menor desviación estándar que los programas del *GC*.

3) Se encontró que en la hipótesis que establece que la media del volumen del programa de los programas hechos por el grupo de experimental es menor que la media del volumen del programa de los programas hechos por el grupo de control, hay una tendencia a favor del 61.83%, esto con un nivel de confianza del 80%. Sin embargo, no hay suficiente prueba estadística que permita aceptar dicha hipótesis.
4) Hay suficiente prueba estadística, con una confiabilidad del 90%, que permite aceptar que la media del nivel del programa de los programas hechos por el grupo experimental es mayor a la media del nivel del programa de los programas hechos por el grupo de control.
5) Se encontró que en la hipótesis que establece que la media del contenido de inteligencia de los programas hechos por el grupo experimental es mayor a la media del contenido de inteligencia de los programas hechos por el grupo de control, hay una tendencia a favor del 34.48%, esto con una confiabilidad del 80%. Sin embargo, no hay suficiente prueba estadística que permita aceptar dicha hipótesis.
6) Hay suficiente prueba estadística, con una confiabilidad del 80%, que permite aceptar que la media del esfuerzo de programación de los programas hechos por el grupo experimental es menor a la media del esfuerzo de programación de los programas hechos por el grupo de control.
7) Hay suficiente prueba estadística, con una confiabilidad del 80%, que permite aceptar que la media del tiempo estimado de programación de los programas hechos por el grupo experimental es menor a la media del tiempo estimado de programación de los programas hechos por el grupo de control.
8) Hay suficiente prueba estadística, con una confiabilidad del 90%, que permite aceptar que la media del nivel del lenguaje de los programas hechos por el grupo experimental es mayor a la media del nivel del lenguaje de los programas hechos por el grupo de control.
9) En conclusión, **la enseñanza de métricas de software sí permite, a los estudiantes, generar código de mejor calidad que si no hubieran recibido tal enseñanza.**

CAPÍTULO 6

CONCLUSIONES Y RECOMENDACIONES

En este capítulo se discuten los resultados del análisis de datos que fueron presentados en el capítulo anterior. Además, se dan las conclusiones y algunas sugerencias para investigaciones futuras.

El objetivo principal del estudio fue: determinar si impartir un curso de métricas de software a los estudiantes de Sistemas Computacionales contribuye para que se genere código fuente de mejor calidad.

Se encontró que impartir un curso de métricas a los estudiantes de Sistemas Computacionales sí contribuye para que generen software con calidad.

La calidad del software se puede ver en que los programas tienen menos impurezas tales como: una menor utilización de operandos y operadores, menos reemplazos innecesarios, más expresiones matemáticas factorizadas, menos operandos sinónimos y/o ambiguos (que se llaman igual pero tienen una función diferente dentro de un programa), menos instrucciones innecesarias, etc.

Si un programa cumple con lo anterior, será mas fácil de entender, más fácil de modificar y por lo tanto será más económico en tiempo, dinero y esfuerzo, para darle mantenimiento.

6.1 DISCUSIÓN SOBRE LAS LIMITACIONES DEL ESTUDIO

Se deben tomar en cuenta las limitaciones del estudio, por lo que sería recomendable confirmar los resultados encontrados en esta investigación, repitiendo el experimento con programas que tengan una mejor dispersión en cuanto a la longitud del programa (*N*) se refiere. Además que se involucren programas con diferentes funciones como: administración de bases de datos, consultas, actualizaciones, funciones matemáticas, menús y otros, no sólo de reportes como lo fue en este estudio.

También se podría aumentar en forma considerable al número de participantes en el grupo de control y el experimental; por decir un número entre 15 a 20.

Por lo tanto, si clasificáramos a los programas por su longitud (*N*) como pequeños (*N*<=**100**), medianos (*N*>**100** y *N*<=**500**) , grandes (*N*>**500** y *N*<=**1500**) y muy grandes (*N*>**1500**), los resultados de esta investigación sólo son validos para programas medianos, más específicamente, cuya longitud oscila entre 90 y 642, y su función sea de reportes.

Por último, para quien desee repetir este mismo experimento, recomiendo que procure que la realización de los programas por parte del grupo experimental sea inmediatamente después que reciban el estimulo experimental, es decir, después que tomen el curso de métricas de software.

6.2 SUGERENCIAS PARA PROGRAMACIÓN ORIENTADA A OBJETOS

El tipo de programación que se utilizó en este estudio es la llamada procedural, hoy, la programación que viene empujando fuerte es la orientada a objetos, que es una

filosofía muy distinta a la programación procedural. Además, investigaciones realizadas han encontrado que las métricas de Halstead son válidas para el lenguaje de Programación Orientado a Objetos C++ [Espinosa,1997]. En una posterior investigación se puede ver si los resultados son válidos para otros lenguajes orientados a objetos como Visual FoxPro.

6.3 SUGERENCIAS PARA LA CALIDAD DEL SOFTWARE POR PUNTOS DE FUNCIÓN.

Otro tema para futuras investigaciones es que sería conveniente ver la medición de la calidad del software a través de los puntos de función [Preessman,1993]. Hay un Grupo Internacional de Usuarios de Punto de Función (International Function Point User's Group UFPUG) que ofrecen información al respecto, para contactarlos accesar: http://www.bannister.com/ifpug/home/docs/comm.htm, otras referencias son: webmaster@softwaremetrics.com, http://www.softwaremetrics.com/esp/fivemajor.htm, http://www.spr.com/library/0funcmet.htm.

Por otro lado, Warren Harrinson del PSU Center for Software Quality Research de Portland State University y Gene Miluk de Denver Metrics Gruop realizan investigaciones para usar La Ciencia del Software como una aproximación a los Puntos de Función para Código Fuente Existente. (Using Software Science as a Proxy for Function Points for Existing Code Assets) para más información en http://www.cs.pdx.edu/~warren/Papers/FP_PR.htm

6.4 OTRAS SUGERENCIAS

1) Se puede medir el desempeño de un grupo de programadores en alguna organización; esto ayudaría a establecer políticas para el estímulo a la productividad. El desempeño puede medirse utilizando el analizador de código para evaluar los programas desarrollados por cada uno de los programadores. Los programas tendrán un nivel del lenguaje diferente debido a las características individuales; entre más alto sea este

valor, los programas tendrán una mejor calidad y por consecuencia la productividad será más alta.

2) Se pueden establecer, utilizando el analizador de código, niveles de lenguaje para el desarrollo de programas, esto es, estándares de programación a los cuales los programadores se deben apegar. Estos estándares se pueden establecer de acuerdo a la función de los programas (reportes, actualizaciones, menús etc.).
3) Se puede evaluar, utilizando el analizador de código, la calidad del software existente, de esta manera se puede crear un registro histórico de la calidad con que se ha desarrollado el software y usarlo como referencia para futuros desarrollos.
4) El analizador de código se puede utilizar como herramienta que sirva de referencia en los concursos de programación que se realizan en algunas instituciones educativas.
5) Se puede repetir este mismo estudio utilizando PROGRESS, ORACLE etc.

ANEXO A

Sistema de Administración Comercial

Para desarrollar la presente tesis se utilizó el **Sistema de Administración Comercial** que se encuentra en un manual de Curso de PROGRESS 6.2 [Curso Progres. P3-3], donde está la explicación, el Análisis y Diseño del Sistema y el modelo entidad-relación. Para mayores detalles refiérase al archivo FILES.DOC que está en el subdirectorio ANEXOA del disquete que acompaña a esta tesis.

Los programas de este anexo son para dar de altas los registros a las bases de datos. Todos los estudiantes que participaron en el experimento (grupo de control y grupo experimental) hicieron los programas del presente anexo. El lenguaje de programación que utilizaron fue FOX PRO 2.6 para Windows.

La intención de que hicieran estos programas fue para que los participantes se adentraran al sistema sobre el cual se basaría el experimento.

Los programas de este anexo están hechos en el lenguaje PROGRESS6.2 porque se facilitó hacerlos en este lenguaje; por lo tanto si desea verlos funcionar deberá correrlos desde el ambiente antes mencionado.

En el disquete que acompaña a la presente tesis encontrará un directorio llamado ANEXOA, en él se contiene 14 archivos: PROVEDOR.P, ORDEN.P, COMPONEN.P, COMPO-F.F, ARTICULO.P, ARTICULO.F, ARTDEPRO.P, ARTDEPRO.F, FILES.DOC, SISTEMA.BI, SISTEMA.DB, SISTEMA.DF, SISTEMA.LG y ANEXOA.DOC. Los archivos con extensión *.P son programas en PROGRESS 6.2, los archivos con extensión *.F son archivos incluidos de los programas *.P y los archivos con extensión *.DOC son archivo en WORD de MSOFFICE.

El sistema tiene 6 tablas: **PROVEEDOR, ARTICULO, ArtDeProv, COMPONENTE, OrdenDeCompra, y DetalleDeOdeC**. (para mayor detalle refiérase al archivo FILES.DOC).

Descripción de los archivos del directorio ANEXOA.

a) PROVEDOR.P : Hace altas de los proveedores en la tabla **PROVEEDOR**.

b) ORDEN.P : Hace las altas de las ordenes de compra en las tablas **OrdenDeCompra** y **DetalleDeOdeC**.

c) COMPONEN.P : Hace altas de los componentes de los artículos en la tabla **COMPONENTE**, tiene el archivo incluido COMPO-F.F.

d) ARTICULO.P : Hace altas de los artículos en la tabla **ARTICULO**, tiene el archivo incluido ARTICULO.F.

e) ARTDEPRO.P : Hace altas, en la tabla **ArtDeProv**, de los artículos que venden los proveedores , tiene el archivo incluido ARTDEPRO.F.

f) FILES.DOC : Contiene una explicación del Sistema de Administración Comercial.

g) SISTEMA.* (.BI, .DB, .DF, y .LG) : Contiene las estructuras de las bases de datos y el contenido de las bases de datos.

h) ANEXOA.DOC : Texto del Anexo A.

Nota.- Para ver estos programas deberá correrlos desde el ambiente PROGRESS 6.2.

ANEXO B

Solución a los Programas del Experimento

Para el desarrollo de ésta tesis, se hicieron diez programas que fueron programados por el grupo de control y el grupo experimental. Antes de pedir a los participantes de la tesis que hicieran estos programas, un servidor los programó personalmente en PROGRESS6.2 con la finalidad de constatar que los programas no fueran a estar más difíciles de lo que los participantes pudieran resolver; se PROGRESS6.2 porque fue el lenguaje que se me facilitó, no hay ninguna otra razón

La intención de éstos programas, como lo menciono anteriormente, es que el grupo de control haga estos programas en FOXPRO2.6 para Windows sin haber tomado el curso de métricas que está en al Anexo E, y el grupo experimental hará éstos mismos programas después de haber el curso.

En el disquete que acompaña a la presente tesis encontrará un directorio llamado ANEXOB, en él se contiene 17 archivos: UNO.P, DOS.P, TRES.P, CUATRO.P, CINCO.P, SEIS.P, SIETE.P, OCHO.P NUEVE.P, DIEZ.P, TRES.SAL, CINCO.SAL, SEIS.SAL, OCHO.SAL, NUEVE.SAL, PROGRAMS.DOC y ANEXOB.DOC.

Descripción de los archivos del directorio ANEXOB.

a) Los archivos con extensión *.P, son programas es PROGRESS 6.2.

b) TRES.SAL : es la solución al programa TRES.P

c) CINCO.SAL : es la solución al programa CINCO.P

d) SEIS.SAL : es la solución al programa SEIS.P

e) OCHO.SAL : es la solución al programa OCHO.P

f) NUEVE.SAL : es la solución al programa NUEVE.P

g) PROGRAMS.DOC : son indicaciones generales para los participante de la tesis para hacer los programas.

h) ANEXOB.DOC : Texto del Anexo B.

Para ver la solución a los programas UNO.P, DOS.P, CUATRO.P, SIETE.P Y DIEZ.P deberá correrlos desde el ambiente de PROGRESS 6.2.

ANEXO C

Analizador de Código Fuente

El instrumento de medición en esta tesis es un programa que se le llama analizador de código. Este analizador fue desarrollado por el Dr. José Luis Martínez Flores cuando obtuvo su título de Maestría en 1994.

El analizador de código fue implementado en lenguaje PASCAL 6.0. Este instrumento de medición fue diseñado originalmente para obtener las métricas de Halstead para cualquier programa hecho en DBASE III y FOXPRO2; ahora bien, la presente tesis está dirigida para programas en FOXPRO2.6 para Windows. Sin embargo, las diferencias entre la versión 2 y la 2.5 para Windows de FOXPRO son más que nada en las ventanas (Jose Javier García Badell 1997) y dado que entre la versión 2.5 y 2.6 de FOXPRO para Windows las diferencias son todavía más pequeñas se considera que el analizador es válido para esta investigación, por lo que la única actualización que se le realizó al analizador fue incluir la instrucción CLOSE DATA en el archivo ARCH_FOX.TXT para que fuera reconocida como operador.

No conforme con lo anterior, el analizador fue probado con varios programas hechos en FOXPRO2.6 para Windows y mostró resultados satisfactorios. Un ejemplo de esta prueba se localiza en el Anexo E de esta misma tesis.

En el disquete que acompaña a la presente tesis encontrará un subdirectorio llamado ANEXOC, ahí están todos los archivos que son parte del analizador de código; para más información acerca del análisis y diseño del analizador remítase a la tesis del Maestría del Dr. José Luis Martínez Flores, 1994.

ANEXO D

Programa hechos por los Grupos del Experimento

En el disquete que acompaña a la presente tesis encontrará un subdirectorio llamado ANEXOD que a su vez contiene los subdirectorios EXPERIME y CONTROL; en el subdirectorio EXPERIME encontrará los programas hechos por los participantes del grupo experimental y en el subdirectorio CONTROL encontrará los programas hechos por los participantes del grupo de control.

Los subdirectorios de cada grupo son A, B, C, D, E y F que se refiere a los alumnos correspondientes.

ANEXO E

Curso de Métricas del Software

Este curso fue impartido únicamente al grupo experimental en un lapso de 6 horas, posteriormente se les pidió que hicieran los programas del Anexo B.

El curso está basado en los capítulos 1, 2, 3 y 20 del libro Ingeniería del Software: Un Enfoque Práctico de Roger S. Pressman, tercera edición. El tema de la ciencia del software de Halstead está en el capitulo 17, sección 17.4 del mismo libro aunque está más ampliamente explicado en la tesis de maestría del Dr. José Luis Martínez Flores.

En el disquete que acompaña a la presente tesis encontrará un directorio llamado ANEXOE, en él se contiene 9 archivos: Cap1.doc, Cap2.doc, Cap3.doc, Cap20.doc, Halstead.doc, Anexoe.doc, Corte2.doc, Corte2.prg, Corte2a.prg.

Descripción de los archivos del directorio ANEXOE.

a) CAP1.DOC : Capitulo # 1, Ingeniería del Software, Roger S. Pressman.
b) CAP2.DOC : Capitulo # 2, Ingeniería del Software, Roger S. Pressman.
c) CAP3.DOC : Capitulo # 3, Ingeniería del Software, Roger S. Pressman.
d) CAP20.DOC : Capitulo # 20, Ingeniería del Software, Roger S. Pressman.
e) HALSTEAD.DOC : Capitulo # 3, Tesis de Maestría del Dr. José Luis Martínez Flores.
f) ANEXOE.DOC : Texto del Anexo E.
g) CORTE2.DOC : Redacción del programa de prueba.
h) CORTE2.PRG : Codificación del programa de prueba, primera versión.
i) CORTE2A.PRG : Codificación del programa de prueba, segunda versión.

Al curso se agregó el ejemplo de un programa hecho en FOXPRO2.6 para Windows el cual se hizo en dos versiones diferentes, es decir, ambos programas hacen exactamente lo mismo; los dos programas fueron sometidos al analizador de código y mostraron resultados significativamente diferentes. Por ejemplo el programa CORTE2.PRG tiene un nivel del lenguaje de 0.387 y el programa CORTE2A.PRG tiene

un nivel del lenguaje 0.656. La diferencia entre ambos programas, en cuanto a codificación se refiere, es poca, por ejemplo CORTE2.PRG tiene 4 instrucciones IF mientras que CORTE2A.PRG tiene 3 instrucciones IF, entre otras diferencias. Se recomienda ver la redacción del programa en el archivo CORTE2.DOC y las codificaciones de los programa, así como someter ambos programas al analizador para ver otras diferencias interesantes.

BIBLIOGRAFÍA

- [Athey,1988] Athey, T.H. y Zmud, R.W. Introduction to Computers and Information Systems, second edition; Scott, Foresman and Company, 1988.
- [Baez-Casselyn,1997] Baez-Ventura: Baez López David J.M. y Casselyn León Ventura. Memoria Técnica/Proceedings. VII CONIELECOMP. Universidad de la Américas-Peubla. IEEE. Congreso Internacional de Electrónica, Comunicaciones y Computadoras. Febrero 1997.
- [Bowman,1990] Bowman, B.J. y Newman W.A. "Software Metric as a Programming Training Tool", J. Systems Software, 1990, Vol 13, pp. 139-147.
- [Crespo,1992] Crespo D. y Vigil r. "Diagnóstico de la Situación Actual de las Unidades de Informática de Informática en la mediana empresa de Cd. Victoria". Tesis de Licenciatura en Computación Administrativa, UAT, Noviembre de 1992.
- [Espinosa,1997] Espinosa de los Monteros Anzaldúa, Xavier. "Métricas de Halstead aplicadas a Lenguajes de Programación Orientados a Objetos". Tesis de Mestría en Ciencias de la Administración con Especialidad en Sistemas. UANL-FIME. San Nicolas de los Garza, NL. México. (Enero de 1997).
- [García-Badell,1997] García-Badel, José Javier. "FOXPRO2.5 para DOS y Windows: A su alcance". McGraw-Hill/Interamericana de España, S.A. enero de 1997.
- [Gutierrez,1998] Gutierrez Alanis, Ma. Guadalupe M.C. "Demanda y Perfil de Profesionistas Solicitados durante el Año de 1997 de las Carreras Ofrecidas por FIME UANL". Secretaría de Planeación y Desarrollo FIME-UANL. Febrero 1998.
- [Halstead,1977] Halstead, M.H. Elements of Software Science, Elsevier North-Holland,1977.
- [Harrinson W,1998] Harrinson, Warren. PSU Center for Software Quality Research. Portland State University.1998. http://www.cs.pdx.edu/~warren/Papers/FP_PR.htm.
- [Hernández,1995] Hernández Sampieri, Roberto, Fernández Collado Roberto y Baptista Lucio Pilar. "Metodologia de la Investigación" McGraw-Hill Interamericana de México, S.A. de C.V. ,1995.

- [McCabe,1976] McCabe, T. "A Software Complexity Measure". IEEE Transaction Software Engineering, Vol. 2, Diciembre 1976, pp. 308-320.
- [McCall,1977] McCall, J., Richards P., y Walters G., "Factors in Software Quality" General Electric, Command and Information Systems, Technical Report 77cis02, Sunnyvale, California,1977.
- [Martínez,1994] Martínez Flores, José Luis. "Métricas de Software en Lenguajes de Cuarta Generación". Tesis de Maestría en Ciencias de la Administración con Especialidad en Sistemas. UANL-FIME. San Nicolas de los Garza, N.L. México. (Marzo de 1994).
- [Mendenhall,1986] Mendenhall, William. Scheaffer, Ricchard L. Wackerly, Dennis D. "Estadística Matemática con Aplicaciones". Grupo Editorial Iberoamericana. 1986.
- [Navarro,1997] Navarro Guerra, Luis Gerardo. "Métricas de Halstead en Lenguajes Manipuladores de Bases de Datos". Tesis de Mestría en Ciencias de la Administración con Especialidad en Sietmas. UANL-FIME. San Nicolas de los Garza, N.L. México (Enero de 1997).
- [Pressman,1993] Preesman, R.S. Ingeniería del Software: Un enfoque práctico, tercera edición; McGraw-Hill/Interamericana de España, S. A., 1993.
- [Pinter,1992] Pinter, L. "Aplique Foxpro", McGraw-Hill/Interamericana de España, S.A., 1992.
- [Ramamurthy,1988] Ramamurthy B. y Melton, A. "A Synthesis of Software Science Measures and the Cyclomatic Number". IEEE Transactions on Software Engineering, Vol. 14, No 8, Agosto 1988, pp. 1116-1121.
- [Shen,1983] Shen, V. Y., Conte, S.D. y Dunsmore, H.E. "Software Science Revisited: A Critical Analysis of the Theory and its Empirical Support", IEEE Transactions on Software Engineering, Vol. SE-9, No. 2, March 1983, pp. 155-165.
- [Sigwart,1990] Sigwart, Ch. D., Van Meer, G.L., y Hansen, J.C. Software Engineering a project oriented approach, Frnaklin, Beedle & Associates, 1990.

- [Weyucker,1988] Weyucker, E.J. "Evaluating Software Complexity Measures", IEEE Transsacctions on Software Engineering, Vol, 14, No 9, September 1988, pp. 1357-1365.
- [Wrigley,1991] Wrigley,C.D. y Dexter, A.S. "A Model for Measuring Information System Size", MIS Quaterly, Junio 1991, pp.245-257.
- [Yourdon,1992] Yourdon, E. Decline & Fall of the American Programmer, Yourdon Press Computing Series, Prentice Hall, 1992.

RESUMEN AUTOBIOGRÁFICO

El Ing. Edgar Danilo Domínguez Vera nació en la ciudad de Poza Rica, Ver. El 9 de abril de 1971. Sus padres son el Sr. Belisario Domínguez Villanueva (†) y Dulce María Vera Ibañez.

Concluyó la carrera de Técnico Programador en el Departamento de Educación Continua de la FIME-UANL en 1989. Terminó la carrera de Ingeniero Administrador de Sistemas, obteniendo título honorífico, en diciembre de 1990 en la F.I.M.E.-U.A.N.L.

Se ha desempeñado como auxiliar del departamento de sistemas y maestro por horas en la F.I.M.E. (1991-1993), como jefe de la carrera de I.A.S. (1993-1996) y como Jefe del Departamento de Informática de Servicios Académicos (1996-a la fecha). Es maestro de tiempo completo en la F.I.M.E. desde 1993.

Pertenece a la Asociación Nacional de Instituciones de Educación en Infomática A.C. donde se ha desempañado como vicepresidente de la región #2: Coahuila, Nuevo León y Tamaulipas (1996-1998) y como Comisario de la Asociación de 1998 a la fecha.

Ha realizado esta tesis titulada "El impacto de la enseñanza de métricas de Software en la implementación de un Sistema Computacional", con la finalidad de obtener el grado de Maestro en Ciencias de la Administración con Especialidad en Sistemas.