InnOvaciOnes de NegOciOs 7(2): 391 - 403, 2010

Identificación de conglomerados (clusters) para implementar objetos de software
(Identification of clusters in order to implement software objects)

Manuel Prieto de Hoyos
Universidad Regiomontana, Monterrey, N.L., México, mprieto@mail.ur.mx

Key Words: Applications, classification, clusters, functionality, objects, software, statistics

Abstract: This work presents a study about an entrepreneurial information commercial system which identifies data clusters which allow recognizing software objects that may be implemented in a entrepreneur framework. This study proposes a classification attending the function of the objects, normality tests are conducted over the data in order to proof the data clusters existence. Finally new steps in the research are suggested to develop the objects found.

Palabras Clave: Aplicaciones, clasificación, conglomerados, estadísticas, funcionalidad, objetos, software

Resumen. Se presenta un estudio de un sistema de información comercial empresarial para identificar conglomerados de datos que permiten reconocer objetos de software los cuales pueden implementarse en un framework empresarial. El estudio propone una clasificación funcional de los objetos y se llevan a cabo pruebas de normalidad sobre los datos para comprobar la existencia de los conglomerados. Finalmente se sugieren nuevas etapas en la investigación para el desarrollo de los objetos resultantes.

Introducción e Hipótesis

Re uso del software

Desde su aparición las computadoras se caracterizaron por requerir programas de cómputo para su funcionamiento, el uso y re uso de dichos

programas de cómputo también denominados software, fue un reto que la industria asumió en plenitud en sus inicios. (Abrams, Stein, 1973)

Los problemas surgieron espontáneamente, los diversos modelos de máquinas requerían diferentes programas ya que su juego de instrucciones era totalmente distinto, una computadora IBM, no podía ejecutar el software de una computadora Digital o CDC, debido a esto, el re uso del software se vio muy limitado (Booth, 1978). A este panorama se agregó el hecho de que cada programa exigía para ser utilizado que los usuarios conocieran a fondo el funcionamiento y forma de uso por lo cual, el entrenamiento del personal tomó especial importancia.

El hecho de que desarrollar software involucra un costo importante de mano de obra calificada, ha alentado a la industria de cómputo a sortear los obstáculos para lograr un eficiente re uso del software. Por ejemplo Sun Systems compañía propietaria del lenguaje Java, ha logrado desarrollar para Java una plataforma que es independiente de la arquitectura del procesador en el que se ejecuta, permitiendo que el software pueda emigrar de una computadora a otra prácticamente sin cambios. (Deitel y Deitel, 2005). Sin lugar a duda una gran facilidad para el re uso de software.

Los Frameworks, la nueva tendencia del re uso del software

Frameworks, es un término que se refiere a una nueva estrategia de solución para requerimientos de proceso de información de las organizaciones, toma el punto de vista de los objetos, no de los procesos, utiliza la teoría de objetos para soportar la solución propuesta. Un framework consiste en una colección de clases, entendiendo por clase la definición genérica de un objeto. Dicha colección permite el re uso de las clases o sea del software, para ello el framework cuenta con el soporte de interfases para su uso. Al utilizar los frameworks por medio de sus interfases, se generan aplicaciones apegadas al contexto.

R. Johnson y E. Foote (1988) definen un framework como “aplicaciones semi completas que pueden especializarse para producir software a la medida”. Pocos estudios se han publicado sobre las ventajas de los frameworks en la producción de software, sin embargo un caso de estudio analizado con rigurosidad científica se presenta en Morisio, M.,

Romano, D. y Stamelos, I. (2002) demostrando y probando las ventajas del uso de esta estrategia.

Hipótesis y Productos del Estudio

En esta investigación se estudia un sistema comercial programado con metodología tradicional, empaquetado, tipo ERP (Enterprise Resource Planning) para determinar si es posible abstraer modelos de objetos que puedan aplicarse en la generación de una colección de objetos que formen un Framework para software empresarial tipo ERP que pueda aplicarse para micro y medianas empresas.

La hipótesis del estudio puede establecerse como sigue: los programas de un sistema comercial empresarial presentan conglomerados (clusters) que obedecen a la funcionalidad de los mismos, estos conglomerados pueden identificarse como: Menús, Actualizadores, Actualizadores Tabulares, Listados, Documentos y Procesos.

Marco Teórico

El paradigma de objetos, también conocido como teoría de objetos, se basa en la modelación del mundo real, tal como los seres humanos lo experimentamos y aprendemos de él, se entiende qué, un objeto no solo es la información que se maneja, sino también las operaciones que se llevan a cabo con dicha información . La intención es formar partes atómicas encapsuladas, de tal manera que dichas entidades conserven integridad en su información (atributos) y en su comportamiento (operaciones) garantizando que una vez probado un objeto no requiere probarse de nuevo y así puede usarse como objeto de biblioteca, esto le permite que pueda intervenir como componente de otros objetos mas complejos en la programación de un sistema. (Martin, J. et al. 1997)

Los objetos se comunican entre sí a través de mensajes, un mensaje es una solicitud de un objeto a otro para que ejecute una determinada operación. Toda la comunicación entre objetos se realiza a través de mensajes.

La programación evolucionó de programación estructurada en los 70's a programación orientada a objetos en los 80's. Las propiedades de la

programación orientada a objetos pueden entenderse mejor a partir de la definición misma de objeto.

Introducción a la Teoría de Objetos.

Atributos

Son las propiedades de los objetos, sus características pueden variar de un objeto en particular a otro y hacen único a cada objeto. Por ejemplo puede haber varios autos pero con diferente color, número de serie, kilometraje recorrido y modelo, estos últimos datos son los atributos del objeto auto. Los atributos describen el objeto, son sus características. Para identificar los atributos hay que poner atención a los adjetivos que describen el objeto.

Un atributo no puede ser alterado o cambiado desde el exterior de un objeto, para que un atributo cambie requiere que se ejecute una operación, esto es, que el objeto reciba un mensaje por medio de un método que solicite el cambio del atributo.

Métodos

Son las operaciones ó procedimientos que el objeto es capaz de ejecutar, en el ejemplo anterior un auto puede viajar, entrar a mantenimiento y cambiar de dueño.

Un método se comunica con el exterior por medio de su capacidad de recibir mensajes de otros objetos, dichos mensajes forman la interfase del objeto con el mundo exterior, como ya se explicó anteriormente un atributo no puede cambiar si no es por el efecto de un método del propio objeto.

Un objeto presenta las siguientes propiedades, tal como lo muestra Martin, J. Y Odell J.J. (1997):

Encapsulación

También denominada encapsulación de la inteligencia, llamada así porque un objeto no se puede abrir sino que tiene una vista externa metafórica correspondiente al mundo real. No es posible cambiar sus atributos directamente estos cambian debido a la ejecución de un método, como ya se explicó anteriormente, volviendo al ejemplo del auto, el kilometraje recorrido de un auto no se puede cambiar si no hay un viaje de por medio, no es válido cambiar el kilometraje recorrido sin asociar el cambio a un viaje .

Autonomía

Un objeto puede ejecutar todas sus operaciones con sus propios recursos y con los recursos que le son enviados en los mensajes a través de la interfase con el mundo exterior. No necesita estar subordinado a la intervención de otros objetos.

Metáfora

Un objeto es un modelo de una entidad que existe en el mundo real, dicha entidad puede ser física o abstracta, desde este punto de vista, un sistema es interpretado como una colección de objetos que forman un modelo de una parte de los procesos del mundo real.

Herencia

Esta propiedad hace posible que un objeto herede de otro sus atributos y procedimientos, es decir que copie de otro objeto sus atributos y procedimientos, por ejemplo, puede existir la definición de vehiculo automotor y esta puede servir como punto de partida para las definiciones de automóvil y auto transporte de carga. Reduciendo en forma notable la cantidad de código a desarrollar, ya que los métodos y los procedimientos comunes a ambas clases derivadas, pueden ser “heredados” de la clase original. Esta propiedad permite que los modelos se asemejen más al mundo real.

Polimorfismo

En ocasiones objetos similares tienen operaciones que aunque son comunes en lo general, sus detalles en lo particular son diferentes, por ejemplo, un auto se enciende y una lámpara se enciende, pero ambos encendidos a pesar de tener una equivalencia lógica, son esencialmente diferentes operaciones.

La Teoría de Objetos permite hacer más eficiente el proceso de desarrollo e instalación de un sistema, enfocándose a los objetos y no a los procesos. Para aplicarla es necesario primero demostrar que dichos objetos existen en el software, localizarlos, identificarlos, y aislarlos con ese objetivo en esta investigación analizamos un sistema con un total de 441 programas computacionales desarrollados con metodología tradicional y utilizando lenguaje FoxPro.

Variables del Estudio

La unidad de medida es un programa o módulo, las variables independientes con que se cuenta y que han sido observadas y medidas son: densidad de datos, densidad de decisión, complejidad dinámica, programador y tipo de módulo. La variable dependiente es tiempo de desarrollo.

Las variables independientes han sido tomadas de los conceptos estudiados por Banker, Davis y Slaughter en 1998:

Definición de las variables

a) Densidad de datos.- Medida que se estima como el número de tablas de la Base De Datos Relacional que el programa utiliza. Es una representación de la complejidad del componente.
b) Densidad de decisión.- Se utiliza en número de instrucciones IF utilizadas dentro del programa y es una medida de la complejidad coordinada del mismo.
c) Complejidad dinámica.- Puede dimensionarse midiendo la cantidad de instrucciones dinámicas, código cambiante dentro de los programas.
d) Programador.- Miembro del equipo de trabajo que desarrolla el módulo.
e) Tipo de módulo.- Los módulos se han clasificado en los siguientes tipos: Menú, Actualizador, Actualizador Tabular, Listado, Documento y Proceso
f) Tiempo de Desarrollo.-Es el tiempo total que se requiere para terminar el módulo

Análisis

Se llevó a cabo el análisis de un sistema tipo ERP, los datos fueron estudiados utilizando el SPSS. Las características del sistema que fue estudiado son las siguientes:

1.- Número de programas: 441
Utilizando una clasificación propuesta por el autor, de acuerdo a su funcionalidad

Los programas están clasificados por tipo de función, esta clasificación es propuesta por este estudio, utilizando los siguientes tipos:

Menú: Este tipo de módulos presentan varias opciones para que el usuario elija una de ellas para su ejecución, después de ejecutarse regresa al mismo menú para ejecutar otra opción.

Actualizador: Un programa de este tipo presenta la información de una entidad para que el usuario elija que hacer con ella: actualizarla, eliminarla o simplemente consultarla. Adicionalmente permite insertar entidades nuevas. Estos programas cuentan con validaciones de los datos exhaustivas para registrar solo información valida.

Actualizador Tabular: Este tipo de módulo presenta la información de una tabla como lo hacen las hojas de trabajo electrónicas, solo que la información es validada para solo permitir información rigurosamente validada.

Listado: Este módulo consta de dos ventanas, en la primera solicita los datos de la consulta que se desea hacer, esto es lo que se utiliza como criterio de selección de la información y en la segunda permite ver dicha información y enviarla a impresión.

Documento: Este tipo de módulo es muy semejante al Listado anterior, solo que en lugar de proporcionar un listado envía un documento como una factura, remisión o nota.

Proceso: Este tipo de programas presentan una venta, solicitan información y llevan a cabo un barrido de una tabla procesando la información solicitada.

El sistema a estudiar cuenta con una distribución como se muestra a continuación:

Actualizadores: 80
Actualizadores. Tabulares: 67
Menús: 42
Listados:175
Documentos: 12
Procesos 65

2.-Tecnología: FoxPro.

3.-Se conoce el programador de cada módulo

4.-Se conoce el tiempo total de desarrollo de cada programa.

5.-Se conocen las variables que utiliza el estudio para cada programa.

Resultados del Estudio y Prueba de la Hipótesis

Pruebas de Normalidad de los datos de tiempo de desarrollo. Se analizó la variable de tiempo de desarrollo de cada módulo, primero se llevó a cabo una prueba de normalidad en todo el conjunto de datos, encontrando que no presentaron distribución normal, como lo muestra el siguiente resultado del SPSS.

One-Sample Kolmogorov-Smirnov Test

		Tiempos hrs.
N		441
Normal Parameters(a,b)	Mean	28.45
	Std. Deviation	18.970
Most Extreme Differences	Absolute	.119
	Positive	.119
	Negative	-.093
Kolmogorov-Smirnov Z		2.495
Asymp. Sig. (2-tailed)		.000

a Test distribution is Normal.
b Calculated from data.

El grado de significancia .000 nos indica que no hay una distribución normal de los datos, sin embargo pudieron detectarse conglomerados (Clusters) por tipo de programa. Presumiendo que esto pudiera indicar la presencia de posibles objetos para cada tipo de programa se llevó a cabo el análisis de cada cluster por separado, encontrando los siguientes resultados:

A continuación se muestra la Prueba de Normalidad de los Actualizadores, en este conglomerado el valor de .004 al ser menor que .005, indica que los datos no se ajustan a la distribución normal sin embargo el valor no es .000

One-Sample Kolmogorov-Smirnov Test

		Tiempos hrs.
N		80
Normal Parameters(a,b)	Mean	24.00
	Std. Deviation	14.707
Most Extreme Differences	Absolute	.198
	Positive	.198
	Negative	-.132
Kolmogorov-Smirnov Z		1.770
Asymp. Sig. (2-tailed)		.004

a Test distribution is Normal.
b Calculated from data.

Prueba de Normalidad de los Actualizadores Tabulares, estos datos si se ajustan a la distribución normal.

One-Sample Kolmogorov-Smirnov Test

		Tiempos hrs.
N		67
Normal Parameters(a,b)	Mean	31.96
	Std. Deviation	18.814
Most Extreme Differences	Absolute	.133
	Positive	.133
	Negative	-.077
Kolmogorov-Smirnov Z		1.085
Asymp. Sig. (2-tailed)		.190

a Test distribution is Normal.
b Calculated from data.

Seguimos con la Prueba de Normalidad de los Menús, este otro conglomerado también se ajusta a la distribución normal.

One-Sample Kolmogorov-Smirnov Test

		Tiempos hrs.
N		42
Normal Parameters(a,b)	Mean	15.98
	Std. Deviation	9.164
Most Extreme Differences	Absolute	.204
	Positive	.204
	Negative	-.192
Kolmogorov-Smirnov Z		1.322
Asymp. Sig. (2-tailed)		.061

a Test distribution is Normal.
b Calculated from data.

Prueba de Normalidad de los Documentos, estos datos siguen también con distribución normal.

One-Sample Kolmogorov-Smirnov Test

		Tiempos hrs.
N		12
Normal Parameters(a,b)	Mean	40.00
	Std. Deviation	22.320
Most Extreme Differences	Absolute	.160
	Positive	.160
	Negative	-.137
Kolmogorov-Smirnov Z		.556
Asymp. Sig. (2-tailed)		.917

a Test distribution is Normal.
b Calculated from data.

A continuación se presenta la Prueba de Normalidad de los Listados, no siguen la distribución normal según el criterio establecido para el valor de la significancia, aunque puede observarse que el valor no es .000.

One-Sample Kolmogorov-Smirnov Test

		Tiempos hrs.
N		175
Normal Parameters(a,b)	Mean	30.04
	Std. Deviation	19.403
Most Extreme Differences	Absolute	.148
	Positive	.148
	Negative	-.139
Kolmogorov-Smirnov Z		1.957
Asymp. Sig. (2-tailed)		.001

a Test distribution is Normal.
b Calculated from data.

Finalmente presentamos la Prueba de Normalidad de los Procesos, los cuales siguen la distribución normal.

One-Sample Kolmogorov-Smirnov Test

		Tiempos hrs.
N		65
Normal Parameters(a,b)	Mean	31.98
	Std. Deviation	22.241
Most Extreme Differences	Absolute	.138
	Positive	.138
	Negative	-.089
Kolmogorov-Smirnov Z		1.116
Asymp. Sig. (2-tailed)		.166

a Test distribution is Normal.
b Calculated from data.

Un resumen de los resultados del análisis de conglomerados son mostrados en la tabla 4.1 sobre los datos para probar la existencia de conglomerados por funcionalidad.

Tabla 4.1 Resultados de la Prueba de Normalidad Kolmogorov-Smirnov, implementada en el SPSS

Conglomerado (Cluster) (Tipo de programa)	Significancia	Resultado ¿Distribución Normal?
Todos los datos	.000	No
Actualizadores	.004	No
Act. Tabulares	.190	Si
Menús	.061	Si
Documentos	.917	Si
Listados	.001	No
Procesos	.166	Si

Fuente: Elaboración propia

Conclusión

Ninguno de los resultados tiene significancia .000 por lo que podemos concluir que efectivamente existe un conglomerado por cada tipo de programa. Adicionalmente observando que hay distribución normal en los datos en cuatro de los seis tipos de programas propuestos nos lleva a concluir que la clasificación puede utilizarse para construir objetos de software basándose en ella.

La aportación de este estudio consiste en el descubrimiento de los conglomerados que indican los patrones básicos de los cuales se forman los sistemas de software. Este resultado tiene aplicaciones tanto en el diseño como en el desarrollo de sistemas.

Los conglomerados encontrados forman los bloques de software o partes funcionales de los sistemas de información comerciales, el descubrimiento de estos conglomerados permite orientar el desarrollo y el diseño de sistemas de una forma rigurosa y científica, utilizando estos

bloques como fundamento del diseño, los sistemas obtenidos serán completos y funcionales, ya que como se observa en el presente estudio estos bloques son los cimientos de los sistemas es por eso que es posible identificarlos estadísticamente.

En desarrollos posteriores esta colección de conglomerados se ha convertido en un grupo de objetos que forman un Framework y que ha sido desarrollado utilizando el lenguaje Java, dicho Framework permite desarrollar software empresarial en tiempo record. El principio es muy simple se utilizan los objetos encontrados en estos conglomerados como bloques de construcción logrando así cubrir las necesidades de proceso y manipulación de información en las organizaciones empresariales de una manera eficiente y práctica.

Referencias

Abrams, M.D. y Stein, P.G. (1973). Computer Hardware and Software. Reading,Massachusetts: Addison-Wesley Publishing Company.

Banker, R.D., Davis, G.B., Salugther, S.A. (Apr, 1998). Software Development Practices, Software Complexity, and Software Maintenance Performance: A Field Of Study. Management Science, 44, 4, 433-450.

Booth, T.L. (1978). Digital Networks and Computer Systems. New York: John Wiley & Sons, Inc.

Deitel, H. y Deitel, P. (2005). How to program in Java 5th edition.. Upper Saddle River, NJ: Prentice Hall.

Hofman, J.D., Rockart, J.F. (1994). Application Templates: Faster, Better, and Cheaper Systems. Sloan Management Review, Fall (36, 1), 49.

Martin, J., Odell, J.J. (1997). Métodos Orientados a Objetos: Conceptos Fundamentales. Nueva York, USA: Prentice Hall.

Morisio, M., Romano, D. y Stamelos, I. (September, 2002). Quality, Productivity, and Learning. IEEE Transactions on Software Engineering, 28 (9), 876-888.

R. Johnson y E. Foote (Junio,1988). Designing Reusable Classes. Journal of Object Oriented Programming, 1 (5).

Mendenhall, W. (1997). Probabilidad y estadística para ingeniería y ciencias. New York: Ed. Prentice-Hall.