Revista de la Facultad de Ingeniería Industrial
Vol. (7) 2: pp. 52-57 (2004) UNMSM
ISSN: 1560-9146 (impreso) / ISSN: 1810-9993 (electrónico)

REUTILIZACIÓN DE SOFTWARE Y SU IMPACTO EN EL COSTO DEL SISTEMA

Recepción: Noviembre de 2004 / Aceptación: Diciembre 2004

(1) Paul Lorena Lazo
(2) Edgar Ruiz Lizama

RESUMEN

El artículo presenta dos programas en C# que utilizan un componente de acceso de base de datos SQL Server 2002, el cual se modifica para ser utilizado con una Base de Datos Oracle 9i, a fin de evaluar la productividad de un desarrollador realizando el análisis comparativo de 2 escenarios: un sistema desarrollado reutilizando software, y otro sin reutilización de software.

Palabras Claves: Reutilización de software. Método aproximativo de costo y productividad. Productividad del desarrollador.

SOFTWARE REUSING AND ITS IMPACT ON THE SYSTEM'S COST

ABSTRACT

This article presents two programs in C# that use an SQL server 2002 data access component, which is modified to be used with a 9i data base, with the purpose of evaluating a developer's productivity, making the comparative analysis of two stages: a system developed reusing software, and another one without software reusing.

Key Words: Software reusing. Cost and productivity approximate method. Developers' productivity.

(1) Ingeniero Industrial. Miembro del Círculo de Investigación y Desarrollo de Software (CIDESOFT), UNMSM.
E-mail: paul_lorena@yahoo.com
(2) Ingeniero Industrial. Profesor del Departamento de Ingeniería de Sistemas e Informática, UNMSM.
E-mail: eruizl@unmsm.edu.pe

INTRODUCCIÓN

El proceso de industrialización del software, exige que los ingenieros y técnicos planteen nuevas alternativas para incrementar la productividad de los desarrolladores y analistas en el desarrollo de sistemas de software.

En este contexto, el estudio evalúa la productividad de un desarrollador realizando una comparación entre un sistema desarrollado reutilizando software, y otro sin reutilización de software. Para ello, en la evaluación se utiliza el Método Aproximativo de Costo y Productividad (COCOMO)

REUTILIZACIÓN DE SOFTWARE

Mientras que los usuarios demandan mayor complejidad y funcionalidad de los sistemas de software; los desarrolladores por su parte, tienen que manejar la complejidad propia de la lógica de negocios, así como la complejidad de la herramienta de programación, con los limitados recursos económicos y de tiempo. En este escenario, la reutilización de software, se constituye como una de las mejores opciones para disminuir los plazos de entrega.

Existen diferentes técnicas de reutilización sistemática de software [4], dentro del esquema de la programación orientada a objetos (POO) uno de ellos es la reutilización de componentes, a través de la creación de clases abstractas, e interfases.

El arquitecto Alexander Christopher, de la Universidad de Oxford, fue quien proporcionó el fundamento teórico que posteriormente consolidó la reutilización de software, por medio de 2 libros publicados en 1977: *«A Pattern Language y A Timeless way of Building»*. Alexander; se refería a construcciones urbanas, alienándolo en estructuras recurrentes a las que denominó patrones[1].

En 1995, Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides [3], plasman los conceptos de Christopher Alexander en el libro *«Design Patterns Elements of Reusable Object-Oriented Software»*. El aporte de este libro es importante ya que estableció los requisitos que debe poseer un patrón de diseño.

PROCEDIMIENTO EXPERIMENTAL

Para evaluar la productividad en el desarrollo de un sistema, reutilizando

el código, es necesario, desarrollar el mismo sistema con la misma funcionalidad pero sin reutilización del código.

Se identifica que uno de los procesos más ampliamente utilizados, es el acceso a una Base de Datos con el fin de obtener o actualizar registros.

Para evaluar la productividad de un componente de software, se establece el siguiente escenario:

- Una Tabla Empleado [EMPLEADO_DATA], con su información básica, en una Base de Datos Oracle 9i.
- Procesos de Inserción en la Tabla Empleado desde un aplicación Web escrita para la plataforma .NET (Framework 1.1.4), utilizando como lenguaje C#.
- Ambas aplicaciones, tendrán la misma interfase gráfica (Front End).
- Utilización del Componente Data Access Application Block V2.0, disponible desde el sitio: *http://www.microsoft.com/downloads/details.aspx?FamilyId=76FE2B16-3271-42C2-B138-2891102590AD&displaylang=en*. Este componente ha sido creado para una Base de Datos SQL Server 2000, se procedió a modificar el código para que sea utilizado en el experimento.

En la Figura 1, se muestra la estructura de la tabla Empleado_Data en Oracle 9i, en la Figura 2 se muestra la interfase de mantenimiento.

Como se indicó el componente del Data Access Application Block V2.0 para SQL Server ha sido modificado para que soporte a la base de datos Oracle. Se adjunta los códigos para la inserción, actualización y eliminación de ambos escenarios (véase Figuras 3, 4, 5, 6, 7 y 8).

En la Figura 9, se puede observar como la reutilización de código, permite encapsular:

- El instanciamiento de funciones como el OracleDataAdapter, OracleCommand.
- El proceso de apertura y cierre de conexiones (open y close).
- La ejecución de las sentencias SQL.

MÉTODO APROXIMATIVO DE COSTO Y PRODUCTIVIDAD (COCOMO)

Para evaluar el impacto en el costo al reutilizar el código, se procede a medir los costos de cada sistema por medio de un método empírico de aproximación planteado por el Dr. Boehm [2]. Este método toma en cuenta 15 factores críticos en el entorno y desarrollo del software (véase Cuadro 1).

Las ecuaciones utilizadas son:

$$PM = 3.2 * EAF * KLOC \wedge 1.05 \quad (1)$$

Donde:
PM: Personal mensual necesario
EAF: Factor de Esfuerzo Ajustado (producto de 15 Factores)
KLOC: Miles de Líneas de código

$$PMF = PM * EAF \quad (2)$$

Donde:
PMF: Personal Mensual Ajustado

$$Productividad = PMF / KLOC \quad (3)$$
$$Semanas = Productividad * 4{,}17 \quad (4)$$

Donde:
Factor 4,17 esta expresado en semanas / mes

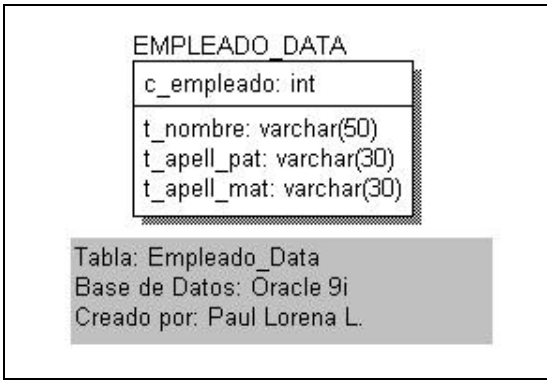

Figura 1. Tabla Empleado

Figura 2. Vista de formulario común

```
void inserta_data_reutiliza()
{
string strConexion = ConfigurationSettings.AppSettings["CnxOracle"];
OracleConnection conn = new OracleConnection(strConexion);
OracleHelper.ExecuteNonQuery(conn, CommandType.Text,
get_sql());
}
```

Figura 3. Reutilización del componente Data Access

```
void insert_data_n_reutiliza()
{
string strConexion = ConfigurationSettings.AppSettings["CnxOracle"];
OracleConnection conn = new OracleConnection(strConexion);
OracleCommand cmd = new OracleCommand(get_sql (), conn);
conn.Open();
cmd.ExecuteNonQuery();
conn.Close();
}
```

Figura 4. Código de la función «insert_data» sin reutilización de código

```
void actualizar_data()
{
string StrConexion = ConfigurationSettings.AppSettings["CnxOracle"];
OracleConnection conn = new OracleConnection(StrConexion);
OracleHelper.ExecuteNonQuery(conn, CommandType.Text,
sql_actualiza());
}
```

Figura 5. Código de la función «actualizar_data» sin reutilización de código

```
void actualizar_data()
{
string StrConexion= ConfigurationSettings.AppSettings["CnxOracle"];
OracleConnection conn = new OracleConnection(StrConexion);
OracleCommand cmd = new OracleCommand(this.sql_actualiza() ,
conn);
conn.Open();
cmd.ExecuteNonQuery();
conn.Close();
}
```

Figura 6. Código de la función «actualizar_data» sin reutilización de código

```
void del_data(int c_empleado)
{
string strConexion = ConfigurationSettings.AppSettings["CnxOracle"];
OracleConnection conn = new OracleConnection(strConexion);
OracleHelper.ExecuteNonQuery(conn, CommandType.Text,
sql_elimina());
}
```

Figura 7. Código de la función «del_data» sin reutilización de código

```
void del_data()
{
string StrConexion = ConfigurationSettings.AppSettings["CnxOracle"];
OracleConnection conn = new OracleConnection(StrConexion);
OracleCommand cmd = new OracleCommand(sql_elimina(), conn);
conn.Open();
cmd.ExecuteNonQuery();
conn.Close();
}
```

Figura 8. Código de la función «del_data» sin reutilización de código

```
108         void insert_data_n_reutiliza()
109         {
110             string strConexion = ConfigurationSettings.AppSettings["CnxOracle"];
111             OracleConnection conn = new OracleConnection(strConexion);
112             OracleCommand command = new OracleCommand( get_sql(), conn);
113             conn.Open();
114             command.ExecuteNonQuery();
115             conn.Close();
116         }


---
113         void inserta_data_reutiliza()
114         {
115         string strConexion = ConfigurationSettings.AppSettings["CnxOracle"];
116         OracleConnection conn = new OracleConnection(strConexion);
117         OracleHelper.ExecuteNonQuery(conn, CommandType.Text, get_sql());
118         }
```

Figura 9. Código de la función «insert_data_n_reutiliza» sin reutilización de código

Duración Estimada Mensual = 2.5 * PMF ^0.38 (5)
Costo en US D = PMF * SM * DS * HD * PH * FA (6)

Donde:
PMF: Personal Mensual Ajustado
SM: Semanas al mes que se trabaja (4,17)
DS: Días a la semana que se trabaja (6 días a la semana)*
HD: Horas al día que se trabajan (7 Horas al día)*
PH: Pago por Hora en dólares (5 US $)*

* Estos factores pueden ser modificados.

En base al muestreo de las líneas de código del sistema en ambos escenarios, se procede a totalizar la líneas de código (LOC: *Lines Of Code*), las que se muestran en el Cuadro 2.

En el Cuadro 3, se procede a ponderar los 15 factores que COCOMO exige para obtener el EAF (*Effort Adjusted Factor*) o Factor Ajustado de Esfuerzo, que se aplica en la ecuación (1). Se observa que para ambos escenarios son los mismos factores a excepción de los factores MODP y TOOL. Varía en ambos casos ya que MODP es un factor que indica si se ha usado técnicas modernas de programación, para el escenario de reutilización es alto HIGH, para el otro escenario (sin reutilización) se considera como LOW.

De igual forma en el factor TOOL, que está referido a la utilización de la herramienta de programación se considera HIGH y LOW para los escenarios con reutilización y sin reutilización respectivamente. Ya que al utilizar componentes, se utiliza toda la potencia y funcionalidad de la programación orientada a objetos, lo contrario pasa en el escenario sin reutilización, en donde se utiliza la programación estructurada típica.

El resultado de los cálculos se muestra en el Cuadro 4. Por otro lado, en el Cuadro 5 se hace una comparación entre ambos escenarios, observándose una

Cuadro 1. Factores de esfuerzo ajustado

				VL	LO	NM	HI	VH	XH
Atributos de Producto	1	**RELY**	Confiabilidad Requerida del software	0,75	0,88	*1,00*	1,15	1,40	1,40
	2	**DATA**	Tamaño de Base de Datos	0,94	0,94	*1,00*	1,08	1,16	1,16
	3	**CPLX**	Complejidad del Producto	0,70	0,85	*1,00*	1,15	1,30	1,65
Atributos de Computadora	4	**TIME**	Restricción en tiempo de ejecución	1,00	1,00	*1,00*	1,11	1,30	1,66
	5	**STOR**	Restricción de almacenamiento	1,00	1,00	*1,00*	1,06	1,21	1,56
	6	**VIRT**	Volatilidad de Memoria virtual	0,87	0,87	*1,00*	1,15	1,30	1,30
	7	**TURN**	Tiempo de Respuesta de Computador	0,87	0,87	*1,00*	1,07	1,15	1,15
Atributos de Personal	8	**ACAP**	Capacidad de Análisis	1,46	1,19	*1,00*	0,86	0,71	0,71
	9	**AEXP**	Experiencia en Aplicaciones	1,29	1,13	*1,00*	0,91	0,82	0,82
	10	**PCAP**	Capacidad del Programador	1,42	1,17	*1,00*	0,86	0,70	0,70
	11	**VEXP**	Experiencia sobre Máq. Virtual	1,21	1,10	*1,00*	0,90	0,90	0,90
	12	**LEXP**	Experiencia en Lenguaje de Program.	1,14	1,07	*1,00*	0,95	0,95	0,95
Atributos de Proyecto	13	**MODP**	Practica Modernas de Programación.	1,24	1,10	*1,00*	0,91	0,82	0,82
	14	**TOOL**	Herramientas de Software	1,24	1,10	*1,00*	0,91	0,83	0,83
	15	**SCED**	Calendario de desarrollo requerido	1,23	1,08	*1,00*	1,04	1,10	1,10

Cuadro 2. Muestreo de línea de código en ambos escenarios

	Inserción	Actualización	Eliminación	LOC
Reutilizando Código	3	3	3	9
Sin Reutilización de Código	6	6	6	18

Cuadro 3. Factores utilizados para el cálculo de costos en ambos escenarios

		Reutilización	Sin Reutilización
Atributos de Producto	**RELY**	0,88	0,88
	DATA	1,16	1,16
	CPLX	1,00	1,00
Atributos de Computadora	**TIME**	1,11	1,11
	STOR	1,00	1,00
	VIRT	1,00	1,00
Atributos de Personal	**TURN**	0,87	0,87
	ACAP	1,00	1,00
	AEXP	1,13	1,13
	PCAP	1,00	1,00
	VEXP	1,00	1,00
	LEXP	1,00	1,00
Atributos de Proyecto	**MODP**	**0,91**	**1,24**
	TOOL	**0,91**	**1,00**
	SCED	1,04	1,04

Cuadro 4. Cálculo del costo del sistema

	Reutilización	Sin Reutilización
KLOC (Miles de Líneas)	0,0090	0,0180
EAF (Esfuerzo Ajustado)	0,9594	1,4365
PM (Personal Mensual)	0,0276	0,0843
PMF(PM Ajustado)	0,0265	0,1210
Productividad	2,9390	6,7243
Semanas	12,2557	28,0404
Duración Estimada Mensual*	0,6287	1,1206
Costos(US $) USD 5 /Hora	0,0265	0,1210

Cuadro 5. Tasas comparativas en ambos escenarios

	Reutilización	Sin Reutilización	%
Plazos (Meses)	0,6287	1,1206	43,89
Costos (US $)	19,3580	88,5797	78,15

reducción en el costo y en el plazo de entrega para el mismo sistema en los dos escenarios planteados.

CONCLUSIONES

Se observa una reducción del 43,89% en el plazo de entrega si se reutiliza software a través de componentes, así como un ahorro del 78,15% en los costos de mano de obra, esto se obtiene por el procedimiento de aproximación de COCOMO.

La reutilización de software, a través de componentes, se constituye como una forma de mejorar la productividad en el desarrollo de sistemas de software.

BIBLIOGRAFÍA

1. Alexander, Christopher. (1977). *A Pattern Language*. Oxford University Press, Inglaterra.

2. Boehm, B.W. (1981). *Software Engineering Economics*. Prentice-Hall, México.

3. Gamma, Erich; Helm, Richard; Johnson, Ralph; Vlissides, John. (1995). *Design Patters Elements of Reusable Object-Oriented Software.* Addison Wesley Publishing Company, Inc. USA.

4. Sommerville, Ian. (2002). *Ingeniería de Software*. Addison Wesley Publishing Company, Inc. USA.