VALIDACION DE MEDIDAS PARA EVALUAR LA MANTENIBILIDAD DE PROGRAMAS SQL[1]

ANTONIO MARTÍNEZ[1,2] y MARIO PIATTINI[1]

[1]GRUPO ALARCOS
DEPARTAMENTO DE INFORMÁTICA
UNIVERSIDAD DE CASTILLA-LA MANCHA
RONDA DE CALATRAVA, 5
13071, CIUDAD REAL. SPAIN.
{mpiattin, amartinez} @inf-cr.uclm.es

[2]EXCMA. DIPUTACION DE CIUDAD REAL
CALLE TOLEDO, 17
13001, CIUDAD REAL (SPAIN)
amartinez@dipucr.es

Resumen. Los entornos de cuarta generación sustituyen cada vez más a los lenguajes de tercera generación como plataforma de desarrollo habitual de sistemas informáticos, por lo que se hace imprescindible controlar la complejidad y mantenibilidad de las aplicaciones desarrolladas en estos entornos. Una forma de realizar este control es mediante la utilización de métricas específicas para los lenguajes de cuarta generación, campo que ha recibido poca atención dentro de la ingeniería del software.En este trabajo se proponen cuatro sencillas métricas para evaluar la mantenibilidad de programas SQL, llevándose a cabo su validación empírica. Esta validación consiste tanto en el estudio de un caso real, compuesto de 143 programas escritos en un entorno de cuarta generación, como en un experimento utilizando el método de Taguchi.

1 Introducción

Muchas organizaciones que utilizan sistemas de información basados en lenguajes de tercera generación, como COBOL, están evolucionando sus sistemas a entornos de cuarta generación, que son más efectivos y utilizan técnicas más modernas [1].

[1] Este trabajo forma parte del proyecto MANTICA, parcialmente financiado por la CICYT y la Unión Europea (1FD97-0168 TIC).

Debido a su creciente difusión, se hace cada día más imprescindible contar con un conjunto de métricas[2] que permita asegurar la calidad de los sistemas desarrollados con este tipo de entornos; especialmente su mantenibilidad, ya que el mantenimiento representa el mayor problema del desarrollo software suponiendo entre el 60 y el 80 por ciento de los costes del ciclo de vida [2],[3]. Como sabemos, las métricas del software son un buen medio para entender, monitorizar, controlar, predecir y probar el desarrollo software y los proyectos de mantenimiento [4] y también pueden ser utilizadas para que los profesionales e investigadores puedan tomar mejores decisiones [5].
Se han definido algunos tipos de métricas para entornos de cuarta generación. Hasta el momento estos trabajos se han centrado en estimar el esfuerzo de desarrollo y la correlación del mismo con el tamaño de un programa L4G [6],[7],[8].
A la hora de definir métricas para programas L4G nos enfrentamos con el problema de la gran heterogeneidad en cuanto a clases de sentencias que pueden componer un programa de este tipo. Nosotros hemos identificado distintos sublenguajes atendiendo a dicha heterogeneidad: sublenguaje de control procedimental, de control visual, de manejo de excepciones, de definición de base de datos, de manipulación de base de datos, de control de seguridad y de control de transacciones [9].
En este trabajo se propone un conjunto de medidas para controlar la mantenibilidad de programas desarrollados en un entorno de cuarta generación, centrándose en el sublenguaje de manipulación de base de datos (sección 2). En la sección 3 se presenta una validación empírica que incluye un caso de estudio y un experimento aplicando el método de Taguchi. Finalmente, en la sección 4 se perfilan las conclusiones y trabajos futuros.

2 Medidas para la validación de programas SQL

Las principales sentencias del sublenguaje de manipulación SQL son: SELECT, INSERT, DELETE y UPDATE. En esta primera aproximación a la definición de métricas para evaluar la mantenibilidad de programas SQL, consideramos cuatro sencillas métricas relativas a estas sentencias:

- **Medida NDS (Número de sentencias SELECT)**
  El número total de sentencias SELECT en un programa SQL.
- **Medida NDI (Número de sentencias INSERT)**
  El número total de sentencias INSERT en un programa SQL.
- **Medida NDD (Número de sentencias DELETE)**
  El número total de sentencias DELETE en un programa SQL.
- **Medida NDU (Número de sentencias UPDATE)**
  El número total de sentencias UPDATE en un programa SQL.

Estas medidas verifican las propiedades de tamaño de Briand et al. (1996), a saber: no negatividad, valor nulo y aditividad de módulos. De esta manera podemos tener una primera validación teórica de las medidas propuestas [10], pero no resulta suficiente ya que debe ser complementada con su validación empírica.

[2] A efectos de esta comunicación utilizaremos de manera indistinta "métrica" y "medida".

3 Validación empírica

Hay que tener en cuenta que la aplicación del paradigma experimental en la ingeniería del software es todavía muy reciente, y que estamos aprendiendo a diseñar experimentos y a extraer conocimiento útil a partir de los mismos [11].
La validación empírica de las medidas propuestas se va realizar sobre un caso real y sobre un caso experimental. El caso real consiste en una aplicación escrita en un entorno de cuarta generación (CA-OpenIngres/4GL). El caso experimental consiste en ocho preguntas combinando las sentencias SQL, según el método de Taguchi. Por lo que conocemos, es la primera vez que se propone aplicar el método de Taguchi en la validación empírica de métricas software.

3.1 Caso de estudio

3.1.1. Caracteristicas generales del sistema

El sistema está compuesto por 143 programas que gestiona el Departamento de Informática de la Excma. Diputación de Ciudad Real. Los programas se desarrollan utilizando una plantilla (existente dentro del departamento de Informática), realizando las entrevistas necesarias con los usuarios que pertenecen al departamento de personal de la propia entidad donde se va implantar el sistema. Los programas que constituyen la aplicación están escritos en un lenguaje de cuarta generación (OpenIngres/4GL) mayoritariamente compuestos por sentencias del lenguaje de manipulación SQL. La aplicación se desarrolló completamente por el mismo grupo de desarrollo, empleándose la misma metodología y utilizando la misma herramienta. análisis Estos factores pueden ser considerados como una constante en el y por lo tanto el grado de fiabilidad del estudio puede ser alto [12].

3.1.2. Recogida de datos

El tiempo (TIEMPO) de mantenimiento incluye añadir funcionalidades para el software (mantenimiento adaptativo) y corregir defectos descubiertos en el sistema (mantenimiento correctivo). Nuestro estudio examinó los datos desde el periodo del comienzo del mantenimiento con la instalación original de la aplicación hasta finalizar la segunda versión de la misma.

3.1.3. Análisis de correlación

Para el test de correlación utilizamos el coeficiente de correlación no paramétrica de Spearman tanto para identificar las relaciones entre la variable dependiente tiempo de mantenimiento (TIEMPO) y las variables independientes (NDS, NDI, NDD y NDU) así como, las interrelaciones que puedan existir entre las variables.
La estadística muestra (tabla 1) que no existe correlación significativa alguna entre las variables independientes (NDS, NDI, NDD y NDU), pero sí se da una correlación

significativa entre la variable dependiente (TIEMPO) y la variable independiente NDS (Coeficiente de correlación de Spearman = 0,803) [12].

		NDS	NDI	NDD	NDU	TIME
Spearman	NDS	1,00	0,41	0,13	0,14	0,80
	NDI	0,41	1,00	0,21	0,09	0,41
	NDD	0,18	0,21	1,00	0,23	0,16
	NDU	0,14	0,09	023	1,00	0,09
	TIEMPO	0,80	0,41	0,16	0,09	1,00
Sig	NDS		0,00	0,13	0,10	0,00
	NDI	0,00		0,01	0,27	0,00
	NDD	0,13	0,01		0,01	0,06
	NDU	0,10	0,27	0.01		0,29
	TIEMPO	0,00	0,00	0,06	0,29	
N	NDS	143	143	143	143	143
	NDI	143	143	143	143	143
	NDD	143	143	143	143	143
	NDU	143	143	143	143	143
	TIEMPO	143	143	143	143	143

Tabla 1. Coeficientes de correlación de Spearman

Esto nos lleva a suponer que la medida NDS es un indicador válido de la mantenibilidad de las aplicaciones desarrolladas con lenguajes de cuarta generación basados en SQL.

3.2 Experimento

Dentro de las diferentes clases de experimentos que se pueden dar en ingeniería del software [10], el más utilizado es el diseño factorial o complejo (que trata con dos o más variables independientes). Fisher utilizó el concepto de diseño factorial para la experimentacion en trabajos de agricultura y en la industria farmaceútica y química [13].
Un diseño factorial completo conlleva un número N de posibles diseños, que es:
$N = L^m$
donde L = Número de niveles para cada factor
y m = Número de factores
Esta relación exponencial, hace muy laboriosa y costosa la experimentación con el diseño factorial completo. Así, por ejemplo, debido a que nuestro experimento tiene cuatro factores NDS, NDI, NDD y NDU y cada uno tiene dos niveles, deberíamos realizar 16 casos distintos para completar el experimento, esto nos llevó a estudiar e investigar otros métodos de experimentación que sean menos laboriosos y costosos , aunque sean métodos diseñados para otros campos y no se hayan aplicado al campo de la ingeniería del software empírica.

3.2.1 Método de Taguchi

Taguchi contribuyó a la disciplina y estructura para diseños de experimentos, obteniendo como resultado una metodología de diseño estandarizado que puede ser fácilmente aplicada por los investigadores. Comenzó a desarrollar nuevos métodos para optimizar los procesos de experimentación en la ingeniería, que han sido aplicados con éxito en el campo de la industria manufacturera.
Los resultados de los experimentos por el método de Taguchi se analizan para alcanzar uno o más de los siguientes objetivos:

1. Establecer la mejor o la condición óptima para un proceso o un producto.
2. Estimar la contribución individual de los factores.
3. Estimar la respuesta bajo la condición óptima.

Taguchi construyó un conjunto de matrices ortogonales para ser utilizadas en diferentes situaciones experimentales. Un ejemplo de matriz ortogonal de siete factores y dos niveles para cada factor (L_8) se muestra en la tabla 2. La matriz está compuesta de 8 filas y 7 columnas. Cada fila representa un caso experimental y los números que figuran en cada fila indican el nivel de cada factor. Las columnas se corresponden con los factores que componen el diseño especificados en el experimento.

NUM EXP.	FACTORES A	B	C	D	E	F	G
1	1	1	1	1	1	1	1
2	1	1	1	2	2	2	2
3	1	2	2	1	1	2	2
4	1	2	2	2	2	1	1
5	2	1	2	1	2	1	2
6	2	1	2	2	1	2	1
7	2	2	1	1	2	2	1
8	2	2	1	2	1	1	2

Tabla 2. Matriz Ortogonal L_8 (2^7)

Cada columna contiene cuatro niveles bajos (representados por el valor 1) y cuatro niveles altos (representados por el valor 2) para cada factor asignado a la columna. Los niveles se combinan de cuatro formas diferentes (1,1), (1,2), (2,1), y (2,2).
La matriz ortogonal facilita el diseño de experimentos, ya que reduce bastante el número de pruebas a realizar. Para el diseño de un experimento se debe seleccionar la matriz ortogonal, asignar los factores a las columnas correspondientes y finalmente describir las combinaciones de los experimentos individuales llamados "condiciones de prueba". La tabla identifica las ocho pruebas necesarias para completar el experimento y el nivel de cada factor para cada prueba que se realiza. La descripción de cada experimento individual está determinado por los números 1 (nivel bajo) y 2 (nivel alto) que aparecen en la fila de la prueba que se realiza. Para el caso anterior, si hubiésemos realizado un diseño factorial completo tendríamos que realizar 2^7=128 pruebas, mientras aplicando el método de Taguchi con 8 pruebas es suficiente sin apreciable pérdida de información.

Nosotros aplicamos el método de Taguchi para conocer el porcentaje de contribución de cada una de las medidas (NDS, NDI, NDD y NDU) para evaluar el tiempo de mantenimiento de la aplicación.

3.2.2 Aplicación del método de Taguchi a las medidas propuestas

En esta sección resumimos el experimento llevado a cabo para validar las medidas. El resultado del caso de estudio nos confirma la intuición lógica de que no existe correlación o interacción entre las variable independientes (NDS, NDI, NDD y NDU) por lo que se puede aplicar el método de Taguchi sin problemas [14].
El objetivo de nuestro experimento es determinar la contribución individual de las medidas que dificultan la entendibilidad de las sentencias de manipulación que componen un progama SQL, la cual influye en la mantenibilidad de los mismos.

3.2.2.1 Diseño

En nuestro caso tenemos cuatro factores (NDS, NDI, NDD y NDU) que los asignamos a las cuatro primeras columnas de la matriz L_8. Las columnas restantes no se utilizan en este experimento. Cada factor tiene dos niveles (1 sentencia o 3 sentencias) (tabla 3).

Descripción variable	NIVEL 1	NIVEL 2
NDS (Número total de SELECT)	1	3
NDI (Número total de INSERT)	1	3
NDD (Número total de DELETE)	1	3
NDU (Número total de UPDATE)	1	3

Tabla 3. Medidas del experimento con sus niveles

3.2.2.2 Sujetos

Los participantes del experimento fueron estudiantes de cuarto curso de la Escuela Superior de la Informática de la Universidad de Castilla-La Mancha. Hasta el día del experimento los estudiantes no sabían lo que iban a hacer. El experimento fue desarrollado por 31 estudiantes, de los que 22 fueron aceptados finalmente al dar respuestas correctas. Para minimizar la variabilidad entre los participantes hemos elegido un grupo de estudiantes del mismo año, y prácticamente los mismos conocimientos y experiencia sobre lenguaje SQL.

3.2.2.3 Material

Las ocho combinaciones de sentencias SQL que requiere el experimento se han entregado en hojas separadas (de la que se muestra un ejemplo en el apéndice A), conteniendo cada una los valores de los niveles que indican las filas de la matriz ortogonal del método de Taguchi (Ej: la prueba número 1 se compone de una sentencia SELECT, una sentencia INSERT, una sentencia DELETE y una sentencia UPDATE; la prueba número 8 se compone de tres sentencias SELECT, tres sentencias INSERT, una sentencia DELETE y tres sentencias UPDATE). La documentación que se acompaña a cada prueba es de aproximadamente treinta páginas de longitud incluyendo la descripción y contenido de las tablas y las consultas.
Los estudiantes deben escribir el tiempo inicial y final así como el resultado de cada caso. El orden de las preguntas es aleatorio para cada sujeto debiendo responder en el orden que se entrega.

3.2.2.4 Análisis de resultados

Para analizar los resultados del método Taguchi utiliza la técnica de estadística de Analisis de Varianza (tabla 4).

	Suma de cuadrados	GL	Media cuadrática	F	Porcentaje de contribución
TIEMPO	613,50	4	153,38	29,93	
NDS	528,13	1	528,13	103,05	83,98
NDI	28,13	1	28,13	5,49	4,47
NDD	36,13	1	36,13	7,05	5,74
NDU	21,13	1	21,13	4,12	3,36
Modelo	613,50	4	153,38	29,93	
Residual	15,38	3	5,13		2,45
Total	628,88	7	89,84		100

Tabla 4. Resultados del experimento

Si comparamos los valores de la tabla 4 con $F_{.05}(1,3)=10,128$ podemos asegurar que:

Como 103,049 > 10,128 podemos decir que NDS sí afecta al tiempo de mantenimiento.
Como 5,488 < 10,128 podemos decir que NDI no afecta.
Como 7,049 < 10,128 podemos decir que NDD no afecta.
Como 4,122 < 10,128 podemos decir que NDU no afecta.
Si observamos el porcentaje de contribución (tabla 4) de cada medida tenemos que el correspondiente a la medida NDS es significativamente grande, no siendo así el que corresponde a las medidas NDI, NDD y NDU.
Podemos concluir que el número de sentencias SELECT de un programa SQL es un indicador sólido de la entendibilidad y por lo tanto influye en la mantenibilidad de los

programas. Las medidas NDI, NDD y NDU podemos decir que no tienen influencia significativa en la mantenibilidad de los programas SQL
NDS puede clasificarse como una métrica dominante, mientras NDI, NDD y NDU se pueden considerar métricas redundantes, siguiendo la terminología de Schneidewind [15].

4 Conclusiones y trabajos futuros

Hay una gran necesidad de medir la calidad de las aplicaciones basadas en lenguajes de cuarta generación. Las medidas pueden ayudarnos para capturar ciertos atributos de calidad de software y estos atributos ser utilizados para construir mejores productos software [16].
Hemos propuesto y validado empíricamente cuatro métricas para la medición de la mantenibilidad del sublenguaje de manipulación de base de datos. Con los resultados obtenidos concluímos que el número de sentencias SELECT (NDS) afecta a la mantenibilidad de los entornos de cuarta generación basados en SQL. Por supuesto que habrá que validar esta suposición en un mayor número de casos y experimentos, y estudiar su generalización a otras aplicaciones desarrolladas en 4GL. Además, las métricas propuestas no son suficientes ya que habrá que profundizar en la sentencia SELECT y elaborar métricas que permitan caracterizarla de forma más precisa. En estos momentos estamos trabajando en otras métricas de la sentencia SELECT, por ejemplo, número de tablas, número de operadores incluídos en la cláusula WHERE, cláusulas, etc.
Además, las métricas del sublenguaje de manipulación de datos no son suficientes para valorar la calidad de los productos desarrollados con lenguajes de cuarta generación, por lo que deberían ser complementadas con otras muchas que permitan caracterizar los distintos sublenguajes que identificamos en los entornos de cuarta generación: visuales, procedimentales, etc.

Referencias

1. Holloway, S. (ed.) (1990). *Fourth-Generation Systems, their scope application and methods of evaluation.* London: Chapman and Hall.
2. Card, D.N. y Glass, R.L. (1990). *Measuring Software Design Quality.* Englewood Cliffs. USA.
3. Pigoski, T.M. (1997). *Practical Software Maintenance.* Wiley Computer Publishing. New York, USA.
4. Briand, L.C., Morasca, S. y Basili, V. (1996). Property-based software engineering measurement. *IEEE Transactions on Software Engineering,* 22(1), 68-85.
5. Pfleeger, S. L. (1997) Experimental Design and Analysis in Software Engineering. *Annals of Software Engineering 1,* 219-253. JC. Baltzer AG, Science Publishers.
6. Dolado, J.J. (1997). A Study of the Relationships among Albrecht and Mark II Function Points, Lines of Code 4GL and Effort. *J. Systems Software,* 37, 161-173.

7. Verner J. y Tate G. (1988). Estimating Size and Effort in Fourth-Generation Development. *IEEE Trans. on Software Engineering.* 15-22.
8. Bourque, P. y Côté, V. (1991). An experiment in software sizing with structured analysis measures. *Journal of Systems and Software 15,* 159-172.
9. Martínez, A. y Piattini, M. (1998). Validation of 4GL metrics. *Proc. of the Software Measurement in Practice, 10th Anniversary Conference.* United Kingdom Software Metrics Association, Londres, octubre 1998, X 1-19.
10. Dolado, J.J. y Fernández, L. (2000). *Medición para la gestión en Ingeniería del Software.* Ed. Ra-Ma, Madrid.
11. Basili, V.R. Shull, F. y Lanubile, F. (1999). Building knowledge through families of experiments. IEEE Trans. on Software Engineering, 25 (4).
12. MacDonell, G., Shepperd, J. y Sallis, J. (1997). Measures for Database Systems: An Empirical Study. *IEEE Software,* 99-107
13. Fisher R.A. (1951). *Design of Experiments.* Edimburgo: Oliver y Boyd.
14. Ranjit, R. (1990). *A primer on the Taguchi method.* Society of Manufacturing Engineers.
15. Schneidewind N.F. (1997). Software metrics for quality control. *Proceedings of the fourth international software metrics symposium.* IEEE Computer Society Technical Council on Software Engineering, 127-136
16. Zuse, H. (1998). *A framework of software measurement.* Walter De Gruyter.

```
:QL

 into control_empleado
),nu_fic,fecha,hora,code_in,contr,estado,code_
))
;
3794X','0959','171298','1600','','ET','C','1',


 control_empleado
 set contr='SM'                HORA INICIO
 id_emp='70568794X' and
 fecha='171298'     and        HORA FINAL
 hora='1600'.
 control_emploado
 where id_emp='70568794X' and
        fecha='171298'    and
       estado='A'.

: * from control_emploado where nu_fic='0959'
echa='171298'
```

fic	fecha	hora	cod-in	contr	status	cod-ce	type