Diseño y Aplicación de Métricas de Software a Lenguajes de Desarrollo de Productos Cliente/Servidor

Carlos Ortiz S., Marcelo Sánchez A., Javier Cañas R., y Marcello Visconti Z.

Departamento de Informática

Universidad Técnica Federico Santa María

Valparaíso - Chile

email: {cortiz, msanchez, jcanas, visconti}@inf.utfsm.cl

Resumen

El presente trabajo presenta una metodología para la evaluación de productos de desarrollo de software a través de la definición de métricas que permitan reflejar su calidad, robustez y eficiencia. Para realizar esta evaluación se desarrollaron cinco aplicaciones *gemelas*.

Las métricas definidas no intentan probar qué tan buena es la calidad de estas cinco aplicaciones desarrolladas, si no que intentan probar qué tan bueno es el lenguaje empleado para su construcción, según la ponderación asignada a cada paquete de desarrollo en base a un conjunto de criterios ponderados y normalizados.

1. Introducción.

Es tradicional en ingeniería medir los recursos invertidos y el posterior desempeño de un diseño, para así poder comparar y mejorar, innovando e incursionando en nuevas técnicas que permitan disminuir los resultados no esperados dando lugar a los resultados deseados. Es por esta razón que antes de invertir en la compra de herramientas de desarrollo se hace siempre necesario evaluar las distintas alternativas para finalmente inclinarse por aquella que destaca ante sus competidoras.

El propósito de este trabajo es definir métricas que determinen la *calidad* de un producto de software destinado al desarrollo de aplicaciones comercializables, especialmente del tipo Cliente/Servidor, cuya evaluación final permita inclinarse por alguno de los productos sometidos a evaluación.

2. Evaluación.

Es necesario evaluar tanto los *kits* de desarrollo como las herramientas de conexión. La evaluación final de la combinación entre cada *kit* de desarrollo y cada herramienta empleada en el acceso a fuentes de datos, es a través de una única magnitud que por sí sola decida, en forma posterior a la evaluación parcial de todas las métricas, qué combinación de productos para el desarrollo es mejor. Para normalizar las distintas unidades de medidas de las métricas, todos los valores resultados de la aplicación de cada métrica deben ser convertidos a una cantidad adimensional, que permita la comparación de productos mediante un único número generado por la operación de los coeficientes finales obtenidos de las distintas evaluaciones.

Por ejemplo, supóngase que las métricas a aplicar son dos: la primera '*x*' cuya unidad de medida es [Segundos] y la '*y*' cuya dimensión es [KiloBytes]. Ambas magnitudes no pueden ser operadas -sumadas por ejemplo-, por ende la comparación entre los elementos que se adjudican dichos valores no puede ser a través de un único valor otorgado por la suma ponderada de '*x*' e '*y*'. Por esto, se hace necesario la conversión de los valores de las métricas '*x*' e '*y*' a nuevas magnitudes adimensionales, que si puedan ser operadas, dado que pertenecen ahora a una misma unidad de medida. La normalización se hace mediante la definición de una escala de valores, que toma el valor en la unidad de medida de '*x*' e '*y*', y lo convierte en una magnitud carente de dimensión para cada métrica. Sin embargo esta solución acarrea otra problemática. Una tabla de conversión o escala, toma una magnitud dimensional a la que asigna un puntaje según el valor numérico adjudicado a la métrica por su evaluación, pero no todas las tablas asignan puntajes en el mismo rango, por lo que implícitamente se está dando relevancia a todas las

mediciones, el cual es inevitablemente arrastrado a la evaluación final. Para entender mejor esta incierta manera de evaluar, es de utilidad plantear un ejemplo: supóngase que las tablas de conversión asociadas a las métricas M1 y M2 son 2.1 y 2.2 respectivamente:

Tabla 2.1: Ejemplo de Conversión de Magnitud expresada en unidades [Dimensión de M1] a puntajes adimensionales.

Valor [Dimensión de M1]	123	88	...	12
Puntaje	23	22	...	0

Tabla 2.2: Ejemplo de Conversión de Magnitud expresada en unidades [dimensión de M2] a puntajes adimensionales.

Valor [Dimensión de M2]	47	90	...	145
Puntaje	4	3	...	0

Supóngase además que el producto sometido a evaluación P1, sólo medido a través de las métricas M1 y M2, obtuvo el puntaje máximo en ambas métricas, es decir 23 y 4 por parte de M1 y M2 respectivamente. Entonces el puntaje final obtenido para el producto P1, calculado como la suma entre ambos valores (23 + 4), resulta ser 27. Pero ambas métricas hicieron su máximo aporte a la puntuación final de P1, por lo tanto ambas deberían contribuir igualitariamente al total, es decir en un 50%; sin embargo M1 aporta el 85.18%, mientras M2 aporta sólo el 14.81%.

Esta irregularidad es solucionada a partir de la definición de un factor para cada una de las métricas denominado *peso inherente*, dado por el cuociente entre el máximo de los puntajes asignados por las métricas, extraído de todas las tablas de conversión, y el puntaje máximo asignado por la métrica para la cual se define su *peso inherente*. Para el ejemplo anterior, el *peso inherente* de la métrica M1 es 23/23, es decir 1, mientras que para M2, el peso inherente es 23/4.

Entones el puntaje asignado por cualquier métrica corresponde al producto entre el puntaje original adimensional, asignado mediante la tabla de conversión, y el factor denominado *peso inherente*.

Fórmula 2.1: Puntaje Final otorgado por una métrica, dependiente del Puntaje Original extraído de la tabla de conversión y de su respectivo peso inherente.

$$PjeMi = h_i * Puntaje\_Original\_Mi$$
$$i = 1..n$$
h_i: Peso inherente de la Métrica Mi

De esta forma, el cálculo del puntaje para P1 proveniente de la métrica M1 es 23*(23/23), esto es 23 y desde M2 es 4*(23/4), es decir 23. La puntuación final para P1, calculado como la suma de los puntajes es 46, donde M1 y M2 hacen ahora un aporte igual equivalente a un 50% cada una.

El puntaje final obtenido por cada paquete o *kit* sometido a evaluación, es la suma ponderada de todas las métricas evaluadas separadamente. Agréguese al ejemplo primero, la suposición de que los productos evaluados son P1 y P2. Entonces los puntajes adimensionales, obtenidos desde las tablas de conversión y de la Fórmula 1, son: 'P1.*x*' y 'P2.*x*' y 'P1.*y*' e 'P2.*y*' para la métrica '*x*' e '*y*' respectivamente. Entonces la magnitud del puntaje final obtenido por P1 y P2, se calcula según la siguiente fórmula:

Fórmula 2.2: Calculo del Puntaje para un *kit* de desarrollo según todas las métricas, dependiente del Puntaje Parcial -otorgado por cada una de las métricas- y del respectivo peso de incidencia.

$PjePi = w_x * Pi.x + w_y * Pi.y$
$i = 1, 2$
w_x: Peso de la Métrica 'x'
w_y: Peso de la Métrica 'y'
$Pi.x$: Puntaje del Producto i - esimo según la métrica 'x'
$Pi.y$: Puntaje del Producto i - esimo según la métrica 'y'

Los pesos *w* empleados para destacar o reducir la importancia e incidencia de cada métrica en el puntaje final para cada producto sometido a evaluación, deben ser un valor igual para cada producto, dado que no sería lícito ni igualitario medir empleando elementos de parametrización distintos.

Si ningún puntaje explica la magnitud obtenida para la métrica que se evalúa, el tasador deberá interpolar, empleando aquellos puntos que más se aproximen al valor asignado, con objeto de conseguir un puntaje para dicha tasación, el cual seguramente es no entero.

La combinación entre un producto de desarrollo de software y una herramienta de acceso a tablas a fuentes de datos, será considerado mejor en tanto presente un puntaje final mayor.

3. Métricas Directamente Evaluables. (Duras)

3.1 Tamaño de la Aplicación Ejecutable.

Medida de la dimensión del producto de software desarrollado expresada en |KiloBytes|, calculada como la suma de todo módulo de tiempo de ejecución (incluyendo también cualquier tipo de librería de tiempos de ejecución DLL y VBX) necesario para que la aplicación pueda funcionar y operar completamente. Si la aplicación se desarrolla

por componentes, es decir, en más de un módulo ejecutable, la magnitud de esta métrica será la suma de los módulos que conforman la aplicación total.

Tabla 3.1.1: de Conversión de Magnitud Tamaño de la Aplicación [KiloBytes] a puntajes adimensionales.

Tamaño [Kb/100]	[0-1]	4	8	12	18	28	36	42	48	56	64	72	80	88	96	[104- [
Ptje	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Peso Inherente -Tamaño de la Aplicación Ejecutable-: 15/15 = 1

3.2 Líneas de código[1] Escritas / Puntos de Función Ajustados.

Corresponde al cuociente entre el número de líneas, que el o los desarrolladores y programadores deberán agregar al código ya generado, para lograr el resultado esperado en la aplicación que se construye, y entre los Puntos de Función Ajustados[2], calculados a partir de los requerimientos. El cuociente intenta reflejar el número de líneas de código necesarias para cubrir un punto de función.

Tabla 3.2.1: de Conversión de Magnitud Líneas de Código Escritas / Puntos de Función Ajustados a puntajes adimensionales.

L.Cód. Escritas/ PFA	[0-1]	2	3	4	5	6	7	8	9	10	13	16	19	25	31	[32- [
Ptje	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Peso Inherente -Líneas de código Escritas / Puntos de Función Ajustados-: 15/15 = 1

3.3 Tiempo Total.

Esta métrica considera el tiempo transcurrido en segundos desde que una consulta sale del computador Cliente, hasta que el mismo despliega por pantalla el último registro de dato proveniente desde el Servidor como parte de la respuesta. Si la medición de esta magnitud se efectúa a través de varias pruebas, el valor en segundos asociado a esta métrica corresponderá al promedio de los tiempos totales asociados a dichas pruebas. Es también importante destacar que todas las pruebas deberán ser realizadas sobre un mismo conjunto de datos, dado que alterar la cantidad de registros, en alguna de las tablas de datos involucradas, entre prueba y prueba, trae directa influencia sobre la magnitud captada para el tiempo total. Si en las pruebas se hacen mediciones con inserciones o actualizaciones, entonces el tiempo total es sólo el que demora la transacción en ser cursada desde su salida desde el computador Cliente hasta que el Servidor ratifica su ejecución como éxito o fracaso.

[1] Cuenta de toda línea de texto que no es: comentario, línea en blanco, línea destinada a testing
[2] Puntos de Función Ajustados, derivado del método *Puntos de Función de IBM-1984*

Tabla 3.3.1: de Conversión de Magnitud Tiempo de Total [Seg] a puntajes adimensionales.

Tiempo Total [Seg]	[0-0.04]	0,7	1,4	2,5	3,5	5	7	9	11	13	15	20	25	30	35	[50- [
Ptje	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Peso Inherente -Tiempo Total-: 15/15 = 1

3.4 Valor de las Herramientas involucradas en el Desarrollo.

El valor asociado a esta métrica, está prestado únicamente por el valor en dólares -sin incluir impuestos- del *kit* de desarrollo dispuesto en Disco Compacto e incluyendo todos sus manuales originales.

Tabla 3.4.1: de Conversión de Magnitud Valor o Costo [US$] a puntajes adimensionales.

Valor [US$/100]	[0-3]	4	5	6	7.8	9.6	11,4	13,2	15	16,8	18,6	20,4	22,2	24	25,8	[30- [
Ptje	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Peso Inherente -Valor de las Herramientas involucradas en el Desarrollo-: 15/15 = 1

3.5 Tiempo de Aprendizaje del Lenguaje de Desarrollo.

Este tiempo se calcula como el promedio de horas necesarias para alcanzar el dominio suficiente, para desarrollar lo exigido en la especificación de requerimientos, en alguna herramienta involucrada en la generación del producto de software final o aplicación ejecutable, suponiendo completo desconocimiento por parte del grupo de programación en el lenguaje empleado. Si en el proceso de programación, quienes participan suman más de uno, entonces el valor asociado a esta métrica es el promedio de los tiempos medidos para cada programador.

Tabla 3.5.1: de Conversión de Magnitud Tiempo de Aprendizaje [Hrs] a puntajes adimensionales.

Tiempo de Aprendizaje [Hrs]	[0-40]	60	70	80	90	100	140	200	[300- [
Ptje	8	7	6	5	4	3	2	1	0

Peso Inherente - Tiempo de Aprendizaje del Lenguaje de Desarrollo-: 15/8 = 1.87

3.6 Tiempo de Programación / Puntos de Función Ajustados.

Medido en horas por puntos de función, representa la inversión de esfuerzo por parte del grupo de desarrollo para construir correcta y cabalmente el número total de puntos de función ajustados expresados en el denominador de esta métrica. El número total de horas involucradas en el proceso de programación, se calcula como la suma de los tiempos individuales invertidos, en el proceso de desarrollo, por cada una de las personas que constituyen el grupo, sin incluir aquí aquel tiempo destinado a resolver dudas respecto del lenguaje gracias a manuales o ayuda en línea.

Tabla 3.6.1: de Conversión de Magnitud Tiempo de Programación / PFA a puntajes adimensionales.

Tiempo de Programación / PFA [Hrs / PFA]	[0-0,2]	0,4	0,7	1	1,3	1,7	2,3	[300- [
Ptje	7	6	5	4	3	2	1	0

Peso Inherente -Tiempo de Programación / Puntos de Función Ajustados-: 15/7 = 2.14

3.7 Tamaño Total de la Herramienta de Desarrollo.

Se mide en Megabytes y expresa el volumen total del paquete o *kit* de desarrollo, instalado completamente. Es parte del paquete: todos los directorios generados en el proceso de instalación, contando incluso aquellos donde residen ejemplos, gráficos y figuras, módulos de ayuda y programas anexos.

Tabla 3.7.1: de Conversión de Magnitud Espacio Físico de Disco [Mbytes] a puntajes adimensionales.

Espacio Físico de Disco [Mbytes]	[0-24]	28	32	36	42	48	60	[70- [
Ptje	7	6	5	4	3	2	1	0

Peso Inherente -Tamaño total de la Herramienta de desarrollo-: 15/7 = 2.14

4. Métricas NO Cuantificables Directamente. (Blandas)

Enfrentados a una métrica no factible de ser cuantificada llanamente, como las del punto 3, se hace necesario la definición y posterior aplicación de una escala de valores, que se detalla para cada requisito, en la que la ausencia absoluta del ítem se castiga con la valoración cero, mientras que la excelencia o presencia indiscutida del ítem se premia con la mayor evaluación de la escala definida.

Si el tasador que hace uso de estas métricas duda de la asignación de un puntaje, dado que la clasificación del producto se encuentra entre dos características de la tabla de evaluación, es lícito entonces, que se asigne un puntaje no entero a la métrica, ajustado siempre a la percepción que el evaluador tiene respecto de esta propiedad.

4.1 Verificador de Consistencia en la Asignación de Tipos de Datos.

Se define el verificador de consistencia como aquella parte del editor de código capaz de detectar la asignación de una variable a otra, donde la primera ha sido generada como un tipo de dato distinto al de la segunda.

Tabla 4.1.1: Generación de puntajes adimensionales dependiendo de la característica -Verificador de_Consistencia en la Asignación de Tipos de Datos-.

Característica	Ptje.
Verificador de consistencia en línea para todo tipo de datos, incluso para tipos definidos por el usuario; exceptuando campos de tablas.	4
Verificador de consistencia para más de uno de los ítems mencionados en el punto anterior.	3
Verificador de consistencia en línea sólo para algunos tipos de datos: String o Logic, o Integer, o Long, o Char, o Valores retornados desde funciones, o Char, definidos por usuario, u otros.	2

No Presenta verificador de consistencia sino hasta el proceso de compilación.	1
No Presenta verificador de consistencia. Los errores se presentan cuando la aplicación es ejecutada.	0

Peso Inherente -Verificador de Consistencia en la Asig. de Tipos de Datos-: 15/4 = 3.75

4.2 Ayuda Sensible al Contexto.

La ayuda sensible al contexto, es la asistencia factible de obtener respecto de una palabra, asociada al lenguaje de programación, desde el editor de instrucciones, con la sola indicación de la misma palabra y la posterior generación de un evento previamente establecido (generalmente presión de teclas 'F_').

Tabla 4.2.1: Generación de puntajes adimensionales según la característica -Ayuda Sensible al Contexto-.

Característica	Ptje.
Presenta ayuda sensible al contexto, mostrando información respecto de cualquier palabra perteneciente al juego de instrucciones y al conjunto de funciones definidas.	2
Presenta ayuda sensible al contexto sólo para sentencias o funciones.	1
No Presenta indicio de ayuda sensible al contexto.	0

Peso Inherente - Ayuda Sensible al contexto-: 15/2 = 7.5

4.3 Detección y Corrección de Errores de Sintaxis.

Se define como la capacidad para corregir palabras y funciones del propio lenguaje, que por efectos de tipeo han sido mal escritas. Por ejemplo en Visual Basic 3.0 la sentencia que cierra un bloque del tipo '*If...Then*' es correctamente escrita '*End If*'. La escritura errada, de esta y otras sentencias, es automáticamente corregida por el editor, para un posterior reconocimiento sin error, esto en tiempos de diseño.

Tabla 4.3.1: Generación de puntajes adimensionales dependiendo de la característica -Detección y Corrección de Errores de Sintaxis-.

Característica	Ptje.
Presenta corrección de palabras para funciones y sentencias.	3
Presenta corrección de palabras sólo para funciones o para sentencias.	2
Presenta corrección de palabras sólo en momentos de compilación.	1
No Presenta indicio alguno de corrección de palabras asociadas al lenguaje.	0

Peso Inherente -Detección y Corrección de Errores de Sintaxis-: 15/3 = 5

4.4 Capacidad para Administrar Defectos de Tiempos de Ejecución.

Esto es la facultad de otorgar a los programas ejecutables capacidad para determinar la secuencia de instrucciones a ejecutar y con ello el destino de la aplicación ante la ocurrencia de un error Por ejemplo Visual Basic 3.0 provee un conjunto de sentencias que permiten desviar la secuencia de ejecución a otra serie de instrucciones mediante las siguientes instrucciones: **On Error Resume** ***next*** **|** ***line***, y **On Error Goto** ***label*** **|** ***line***.

Tabla 4.4.1: Generación de puntajes adimensionales dependiendo de la característica -Capacidad para Administrar Defectos de Tiempos de Ejecución -.

Característica	Ptje.
Provee instrucciones y/o funciones que permiten desviar el curso de ejecución a un grupo no necesariamente secuencial de instrucciones, ante la ocurrencia de errores.	1
No Presenta instrucciones ni funciones que detecten de manera automática la presencia de errores de ejecución.	0

Peso Inherente -Capacidad para Adm. Defectos de tiempos de Ejecución-: 15/1 = 15

4.5 Prestación de Servicios de Traza de Ejecución.

La elevada complejidad de programación de algunas aplicaciones exige la presencia de elementos de ayuda para el desarrollador, uno de estos es el la traza en tiempo de ejecución o *Debugger*.

Tabla 4.5.1: Generación de puntajes adimensionales dependiendo de la característica -Prestación de Servicios de Ruteo-.

Característica	Ptje.
Presta un extendido servicio de traza; pasa repetidas veces por un grupo de instrucciones a voluntad del usuario, permitiendo la impresión de cualquier variable involucrada en el programa. Permite la introducción del *debugger* a cualquier módulo llamado desde el conjunto de instrucciones que se supervisa.	2
Presta pobres servicios de traza; pasa repetidas veces por un grupo de instrucciones, a voluntad del usuario, prestando información respecto del comportamiento del programa, pero no presta opciones para conocer valores de variables involucradas en el programa.	1
No Presta servicios de traza.	0

Peso Inherente -Prestación de servicios de Ruteo-: 15/2 = 7.5

4.6 Calidad de la Ayuda.

La presencia de ayuda en cualquier forma presta especial importancia cuando se trabaja con productos de desarrollo de relativa complejidad. El puntaje de esta métrica debe es marcado por la presencia o ausencia de demostrativos, tutoriales, ejemplos y referencias a instrucciones similares en la ayuda.

Tabla 4.6.1: Generación de puntajes adimensionales dependiendo de la característica -Calidad de la Ayuda-.

Característica	Ptje.
Presenta apoyo solvente respecto del alcance y del detalle de instrucciones, funciones y sentencias del lenguaje, además agrega más de un ejemplo por cada sentencia o función cuestionadas. Elementos presentes en forma fuerte son editores de ayuda Hipertexto y referencias entre comandos relacionados. Incorpora tutoriales explicativos respecto de la operación del paquete de software y de sus potencialidades relevantes.	4
Presenta apoyo solvente respecto del alcance y del detalle de instrucciones, funciones y sentencias del lenguaje, además agrega ejemplos que contribuyen a formar una clara idea del uso de las sentencias cuestionadas. Editores de ayuda que hacen uso extensivo del concepto Hipertexto y referencias cruzadas entre comandos relacionados. Incorpora tutoriales breves respecto de la operación del paquete de software.	3
Presenta asistencia moderada respecto del alcance y del detalle de instrucciones, funciones y sentencias del lenguaje. Se hace acompañar de algunos ejemplos pobres respecto de las mismas sentencias. Presencia leve de editores con búsqueda automática de comandos relacionados, Hipertexto poco solvente.	2
Presenta una pobre descripción respecto de instrucciones, funciones y sentencias del lenguaje, pero no presta	1

ejemplos respecto de las mismas. Editores de ayuda son del tipo 'Browse', para búsqueda manual, ausencia de referencias a palabras de comandos ligados.	
No Presta ayuda de tipo alguno; ni ejemplos, ni tutoriales de aprendizaje.	0

Peso Inherente -Ayuda de buena calidad-: 15/4 = 3.75

4.7 Amortiguación Automática de Errores de Tiempo de Ejecución.

Con la ocurrencia de un error de tiempo de ejecución, es tradicional que al no disponer de una instrucción dispuesta a su captura y posterior traslado del control a un nuevo grupo de instrucciones, la aplicación entera se cierre, poniendo con ello fin a la ejecución del programa. La captura automática de errores y el posterior despliegue de mensajes que revelen la descripción de la ocurrencia, sin poner con ello fin a la ejecución de la aplicación se denomina *amortiguación* del error.

Tabla 4.7.1: Generación de puntajes adimensionales dependiendo de la característica -Amortiguación Automática de Errores de Tiempo de Ejecución-.

Característica	Ptje.
Corrige cualquier tipo de error, incluyendo asignaciones de tipos incompatibles. Luego de la ocurrencia de un error, la aplicación indica el error mediante un mensaje, para algunos tipos de incorrecciones.	3
Corrige o *amortigua* en forma automática algunos tipos de error, por ejemplo: referencias a índices inexistentes ó lectura de registros en archivos cuyo cursor se encuentra en valor EOF. La ocurrencia de cualquier tipo de error no se hace acompañar de un mensaje.	2
Presta funciones y sentencias para la captura y desvío del control en caso de error, pero lo anterior debe ser escrito (programado).	1
No Presta funciones ni sentencias para la detección de ocurrencias de error, ni para el desvío del control.	0

Peso Inherente -Amortiguación Automática de Errores de Tiempo de Ejecución-: 15/3 = 5

5. Productos Evaluados.

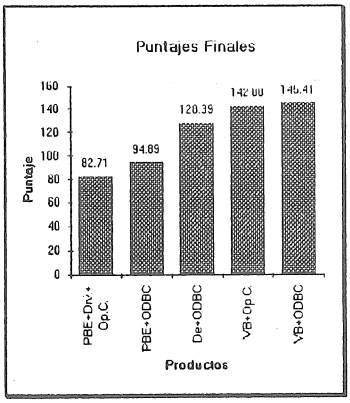
Para someter a evaluación los productos, se ha hecho necesario la definición de una pequeña situación ficticia de 25 Puntos de Función Ajustados, en la que se hacen expresos requerimientos respecto de la arquitectura a emplear, esto es: el Servidor corresponde a Sybase 10 montado sobre Unix, siendo el protocolo de red empleado TCP/IP, mientras por el lado del Cliente se tiene una plataforma de Windows 3.11, generando una capa de red a través del producto Windows Socket.

Los valores asociados a las métricas, la puntuación adimensional y las combinaciones de productos evaluados son presentadas a continuación.

Tabla 5.1: Resumen de Puntajes para las cinco combinaciones evaluadas, considerando peso igual a 1 para cada métrica.

	VB+ODBC[3]	PBE+ODBC[4]	De.+ODBC[5]	VB+Op.C.[6]	PBE+Drv + Op.C.[7]
Tamaño Ejecutable	7,17	6,52	8,41	8,95	0,00
Lin. Cód. Escritas / PFA	4,52	4,94	6,44	1,28	4,94
Tpo. Total	8,22	7,35	2,48	7,59	4,22
Valor de Herramienta de Desarrollo[8]	14,75	0	12,05	14,75	0
Tpo. de Aprendizaje del Leng. de Desarrollo	8,22	10,28	11,03	8,22	10,28
Tpo. de Prog. / PFA	5,24	6,84	7,98	4,81	6,84
Tamaño Herramienta de Desarrollo	9,80	5,84	0,00	9,82	3,31
Verif. de Consistencia en Asignación	9,37	3,75	3,75	9,37	3,75
Ayuda Sensible al Contexto	15,00	0,00	15,00	15,00	0,00
Detección y Corrección de Errores de Sintaxis	15,00	5,00	5,00	15,00	5,00
Adm. de Defectos en tiempos. de ejecución	15,00	15,00	15,00	15,00	15,00
Prestación de servicios de Ruteo	15,00	15,00	15,00	15,00	15,00
Calidad de la Ayuda	13,12	9,37	11,25	13,12	9,37
Amortiguación automática de errores de Ejecución	5,00	5,00	15,00	5,00	5,00
∑ Puntajes	145,41	94,89	128,39	142,88	82,71

Gráfico 5.1: Resumen de Puntajes para las cinco combinaciones de productos evaluados.



Del anterior gráfico es inmediato concluir que en el mismo escenario donde las mediciones fueron efectuadas, esto es un Servidor Sybase 10 montado sobre UNIX, en una red TCP/IP, y un PC Cliente 486 DX2-

[3] Microsoft™ Visual Basic 3.0 más Microsoft™ ODBC - 1
[4] Power Builder Enterprise 4.0 más Microsoft™ ODBC - 1
[5] Borland™ Delphi Estándar más Microsoft™ ODBC - 1
[6] Microsoft™ Visual Basic 3.0 más Microsoft™ Open Client VBSQL.VBX
[7] Power Builder Enterprise 4.0 más Drivers Nativos para Sybase 10 más Open Client Sybase 10
[8] Derivado de cotización al 25 de Septiembre de 1995

66Mhz-8 Mb RAM, la peor combinación de lenguaje más herramienta para el acceso a fuentes de datos es PBE+Drv + Op.C.[9] -Power Builder Enterprise 4.0 con sus *drivers* nativos para Sybase junto a Open Client Sybase 10- con un puntaje final igual a 82,71 [unidades]. Mientras el otro extremo, el ganador, es Visual Basic 3.0 más ODBC, con un puntaje adimensional igual a 145,41 [unidades]. (siempre y cuando los pesos otorgados a cada métrica sean igual a 1)

Es necesario destacar que este trabajo no pretende evaluar los productos comerciales que se mencionan, sino sólo mostrar la metodología. Cada equipo de desarrollo tiene sus particularidades que se deben reflejar en las distintas ponderaciones que se asignen a cada métrica.

Conclusiones y Proyecciones Futuras

Antes de hacer la elección de un lenguaje y una herramienta de acceso a fuentes de datos, para enfrentar el proceso de programación, cabe otorgar tiempo a la evaluación de los productos disponibles y factibles de ser empleados, esto dado las características particulares y distantes logradas al combinar ambos elementos, que en general potencian distintos aspectos y servicios, cuya elección final y correcta aplicación puede significar un notorio ahorro en la fase de programación.

Futuras métricas, destinadas a reflejar calidad y eficiencia de los paquetes de programación, pueden resultar de la apreciación de nuevas características incluidas en los últimos productos de programación lanzados al mercado y de las múltiples necesidades de los productores de software, generadas siempre en torno al proceso de programación, resultando con ello un listado de nuevas instancias de evaluación, que esta metodología puede incorporar sin ninguna salvedad.

Referencias Bibliográficas

[1] John Cratty
Delphi: Visual programming in action
IEEE Computer, Septiembre de 1995

[2] Rosemary Cafasso
Users build Client/Server benchmarks.
ComputerWorld, 26 de Septiembre de 1994.

[3] Jeri Edwards, Dan Harkey y Robert Orfali
Intergalactic Client/Server computing.
Byte, Abril de 1995.

[4] Steve Moore
Client/Server lacks correlation tools.
ComputerWorld, 31 de Octubre de 1994.

[5] Allan J. Albrechr y John E. Gaffney, Jr.

[9] Power Builder Enterprise 4.0 más Drivers Nativos para Sybase 10 más Open Client Sybase 10

Software Function, Source Lives of Code, and Development Effort Prediction: A Software Science Validation
IEEE Transactions on Software Engineering, Vol SE-9, N° 6, Noviembre de 1983

[6] Capers Jones
Applied Software Measurement, McGraw - Hill, 1991