SciVerse ScienceDirect
Disponible en www.sciencedirect.com

00 (2012) 1–11

R I A I
www.elsevier.es/RIAI

Lenguaje de Marcos Conceptuales −*LMC*−: Acercando el Diseño y la Implementación

Bolaños Sandro J.[a,*]

[a] *Facultad de Ingeniería, Universidad Distrital, Kra 7 no 40-53, Bogotá, Colombia.*

Resumen

Este artículo presenta el Lenguaje de Marcos Conceptuales −*LMC*−, su objetivo es cerrar la brecha entre los lenguajes de programación y los lenguajes de diseño, empleando el mecanismo de la esquematización, este propone cambiar la complejidad de la sintaxis de los lenguajes de programación y la complejidad de la diagramación por la facilidad de ensamble y anidamiento de marcos o bloques conceptuales a manera de lego, se presenta las posibilidades que brinda LMC como un leguaje mas próximo a la resolución de problemas empleando un vocabulario computacional y científico que se hace transparente al usuario, se plantean comparaciones e integraciones con lenguajes como java y UML, se proponen métricas y se hace una implementación de la plataforma en lenguaje java.

Palabras Clave: Lenguaje, esquema, métricas, contextualización, abstracción, vocabulario, sintaxis, semántica.

1. Introducción

Es necesario contar con nuevos lenguajes que se acerquen de manera lógica al lenguaje humano o que por lo menos estén en un nivel de abstracción superior, que haga transparente muchos de los conceptos computacionales que son complejos y poco intuitivos. Existen muchos lenguajes de programación y de modelamiento. Cada uno ofrece una forma de representación a través de principios y sintaxis particulares, adoptando mecanismos sobre los cuales basan sus abstracciones. Sin embargo, estos se preocupan muy poco por atender problemas como:

- Representación sencilla e intuitiva
- Hablar un lenguaje menos computacional
- Disminución de la curva de aprendizaje del lenguaje
- Pocos mecanismos, pero infinitas abstracciones
- Trazabilidad directa
- El mismo lenguaje se convierta en el vehículo de la abstracción

Un lenguaje es más poderoso en la medida en que logre ser más sencillo, robusto y completo. Esto se consigue ocultando sus niveles de complejidad, a través de capas, las cuales en principio deben ser al menos dos.

Un primer nivel formal que soporte y extienda los mecanismos que ofrecen ventajas de abstracción ya demostradas por los desarrollos a los que se han llegado, como por ejemplo: la encapsulación, la seguridad, la generalidad, el reuso entre otros (Budd, 1994). No sin antes detenernos a reflexionar si estos mecanismos son propios de los paradigmas de programación o posibles efectos colaterales de los mismos modelos de abstracción. Un segundo nivel concreto y de aplicación directa cuyo uso no implique un alto grado de conocimiento de los problemas computacionales, y que enfoque más su esfuerzo hacia un modelamiento cada vez más aproximado a la realidad, *"Los problemas significativos que enfrentamos no pueden resolverse al mismo nivel de pensamiento en que estamos cuando los creamos"*. Albert Eistein. (Thorpe, 2000).

Para tratar estos planteamientos, el artículo presenta el lenguaje, su gramática, la comparación entre esquema y diagrama, principios y métricas, cerrando con la implementación de la plataforma (en Java), junto con casos de estudio, conclusiones y trabajos futuros.

2. Lenguaje de Marcos Conceptuales LMC

El lenguaje de marcos conceptuales LMC, es un lenguaje de modelamiento enfocado en la abstracción y contextualización del conocimiento. Ver figura 1.

*Autor en correspondencia.
Correo electrónico: sbolanos@udistrital.edu.co (Bolaños Sandro J.)

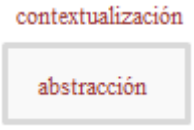

Figura 1: Contextualización y abstracción en LMC

Cuando se necesita comunicar una idea, se acude al lenguaje ya sea este hablado o escrito, la historia del hombre su tradición y cultura han permanecido generación tras generación gracias al lenguaje. Tesoros como la piedra roseta (Budge and Wallis, 1857-1934), develaron todo un pasado impregnado en pictogramas; las pinturas del arte rupestre nos hablan de la vida de nuestros antepasados, en imágenes que constituyen un idioma meno elaborado pero no por ello menos expresivo; el pentagrama en la música, la pintura en el arte, las letras en la literatura o las matemáticas en la ciencia, muestran como recurrentemente hacemos uso de algún tipo de lenguaje. El leguaje puede ir desde una expresión informal, plasmada en una imagen, hasta la expresión rigurosa representada en un vocablo o una estructura sintáctica (Chomsky, 2004). A pesar de ser más entendible una imagen que una letra del alfabeto, no escribimos como los antiguos egipcios, pero eso no significa que se puedan usar figuras o esquemas más intuitivos para expresarnos.

La propuesta detrás de LMC es explotar el poder del lenguaje de manera formal con el poder de la esquematización simbólica, esta dupla propone un bloque de construcción que configura el mecanismo de abstracción y contextualización fundamental del conocimiento modelado en LMC y que se puede definir como *Marco Conceptual*. Resulta muy importante establecer un bloque de construcción pues a través de ellos es posible expresar ideas, así como configurar conjuntos conceptuales más robustos.

3. Abstracción y contextualización

LMC propone dos conceptos, la abstracción y la contextualización. Cuando se plantea el mecanismo de la abstracción se esta ante el esquema quizá más conocido para hacer razonamiento acerca de un determinado fenómeno, la abstracción permite extraer las características esenciales, esto acudiendo a mecanismos cognitivos ya cimentados en nuestro cerebro es decir nuestros paradigmas (Kuhn, 1971), valores principios y comportamientos. Podríamos decir que este es un mecanismo de introspección es decir un mecanismo de búsqueda en nuestro mismo acervo de conocimiento. Por otro lado cuando se establece el concepto de contextualización se trata de acudir a un esquema de observación del entorno tratando de encontrar las repuestas en fenómenos externos a los que inicialmente tenemos como los paradigmas ya adquiridos. El mecanismo que se utiliza podría establecerse como la inmersión que en contraste con el de introspección, busca el razonamiento acera de los fenómenos acudiendo a estructuras externas generalmente elaboradas y que aceptamos como validas o no y con lo cual es posible elaboran una mejor apreciación.

Estos dos mecanismos de razonamiento complementan de manera fundamental la construcción de marcos conceptuales y podría establecerse como efectos colaterales del concepto de los mismos marcos conceptuales. La contextualización y la abstracción son la llave perfecta de consolidar el conocimiento pues contrasta el conocimiento individual con el conocimiento colectivo, permitiendo establecer modelos más robustos.

Clásicamente en modelos como por ejemplo el modelo estructurado o el orientado a objetos incluso los mismos lenguajes declarativos fundamentan los mecanismos en la abstracción (Tucker and Noonan, 2002), perdiendo en gran medida el potencial de la inmersión en una problemática más amplia que permita modelar, con mas precisión los fenómenos reales.

4. Gramática en LMC

A Continuación se definen los conceptos de gramática, derivación (producción) y leguaje, fundamentales para la formalización de LMC.

Una gramática con estructura de frases $G = (V, T, S, P)$ consiste en un vocabulario V, un subconjunto T de V formado por los elementos terminales y un símbolo inicial S de $V - T$ y un conjunto P de producciones. El conjunto $V - T$ se denota por N. Los elementos de N se llaman elementos no terminales. Toda producción debe contener al menos un elemento no terminal en su lado izquierdo (Rosen, 2004). Por otro lado un vocabulario V (o alfabeto) es un conjunto finito y no vacío, cuyos elementos se llaman símbolos, una palabra sobre V es una cadena finita de elementos de V. La palabra vacía o cadena vacía denotada por λ es la cadena sin símbolos. El conjunto de todas las palabras sobre V se denota por V^*. Un lenguaje sobre V es un subconjunto de V^* (Rosen, 2004).

Otra definición fundamental es la de derivación: sea $G = (V, T, S, P)$ una gramática con estructura de frases. Sean $w_0 = lz_0r$ (esto es la concatenación l, z_0 y r) y $w_1 = lz_1r$ sobre V. Si $z_0 \rightarrow z_1$ es una producción de G, decimos que w_1, se deriva directamente de la w_0 (o que es directamente derivable), y escribimos $w_0 \Rightarrow w_1$. Si $w_0, w_1, ..., w_n$ son cadenas sobre V tales que $w_0 \Rightarrow w_1, w_1 \Rightarrow w_2 ... w_{n-1} \Rightarrow w_n$ decimos que w_n es derivable o se deriva de w_0 y se denotará $w_0 \overset{*}{\Rightarrow} w_n$. La secuencia de pasos utilizados para obtener w_n a partir w_0 se llama derivación (Rosen, 2004).

Finalmente se debe definir el lenguaje generado por G (o el lenguaje de G), denotado por L(G) como el conjunto de todas las cadenas terminales que se derivan del estado inicial S, ecuación 1. (Rosen, 2004).

$$L(G) = \{w \in T^* \mid s \overset{*}{\Rightarrow} w\} \tag{1}$$

4.1. *Producciones de LMC en BNF*

La forma Backus Naur fue inicialmente creada para definir la estructura sintáctica del lenguaje de programación algol60 (Teufel and Schimidt, 1995). BNF permite definir la estructura sintáctica del lenguaje, para LMC se tiene las siguientes estructuras sintácticas:

⟨marco conceptual⟩ ::= ⟨marco⟩⟨concepto⟩
⟨marco⟩ ::= ⟨frontera cerrada⟩|⟨frontera abierta⟩|
⟨frontera semicerrada⟩
⟨concepto⟩ ::= ⟨criterio⟩|⟨arquetipo⟩⟨criterio⟩|
⟨arquetipo⟩⟨ criterio⟩
⟨separador⟩⟨cuerpo⟩
⟨criterio⟩ ::= ⟨definitorio⟩|⟨ indagatorio⟩|
⟨comprobatorio⟩|⟨elaborativo⟩
⟨definitorio⟩ ::= ⟨variable⟩|⟨nunciado ⟩|⟨texto⟩
⟨indagatorio⟩ ::= ⟨utilizacion logica⟩⟨?⟩
⟨comprobatorio⟩ ::= ⟨accion de verificacion⟩⟨!⟩
⟨elaborativo⟩ ::= ⟨extension de usuario⟩
⟨arquetipo⟩ ::= λ|⟨ propiedad⟩⟨ nombre⟩|⟨propiedad⟩
⟨nombre:⟩⟨categoria⟩|⟨propiedad⟩
⟨nombre⟩ : ⟨categoria⟩(⟨interaccion⟩)
⟨separador⟩ ::= ⟨secuencial⟩|⟨paralelo⟩
⟨cuerpo⟩ ::= λ|⟨marco conceptual⟩|⟨cuerpo⟩
⟨marco conceptual⟩

5. LMC como lenguaje

LMC está conformado, por un vocabulario o elemento conceptual, la sintaxis o bloque conceptual y la semántica o método conceptual. Ver figura 2.

Figura 2: LMC como lenguaje

5.1. *Vocabulario de LMC*

LMC se conforma de dos conceptos el marco y el concepto. Ver figura 3.

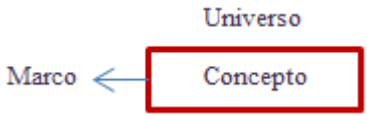

Figura 3: marco

Marco: es la frontera que permite separar un concepto específico del universo del discurso contextualizándolo apropiadamente según el dominio que se pretenda establecer.

Concepto: Configura el dominio del conocimiento que se desea extraer del universo del discurso. Un concepto está constituido por un arquetipo, un criterio, un separador y un cuerpo.

Con las definiciones marco y concepto es posible diseñar por completo el esquema ver figura 4.

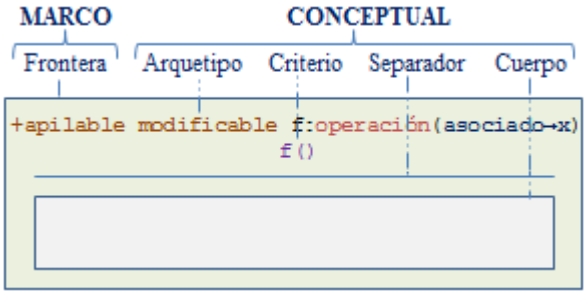

Figura 4: Marco conceptual

Arquetipo: El arquetipo establece las propiedades, la identificación, la categoría y la interacción que va a tener el concepto con el universo del discurso. Ver figura 5.

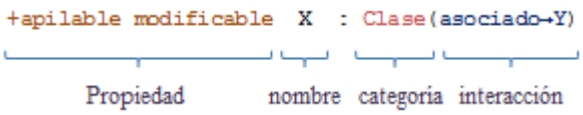

Figura 5: Arquetipo

- Las propiedades permiten establecer rasgos de seguridad, almacenamiento y las formas de modificación del concepto.
- Un concepto puede ser anónimo o tener un nombre caso en el cual permite establecer una particularidad “instancia”
- La categoría permite definir si el concepto está en el dominio del leguaje objeto o en el dominio del metalenguaje
- Con la interacción el arquetipo permite establecer de manera explícita relaciones, conexiones y enlaces.

Criterio: Configura el concepto estableciendo, su definición, posibles formas de indagación, comprobación, y elaboración.

Separador: Establece el límite entre criterio y cuerpo, incluso en el cuerpo mismo también separa un marco conceptual de otro. El separador es importante para demarcar el criterio. El separador establece también como será interpretado el cuerpo. Básicamente hay dos interpretaciones el modo y el tipo. El modo establece si la interpretación es paralela o secuencial en tanto que el tipo establece si se interpreta como directo o recursivo el o los marcos contenidos en el cuerpo.

Cuerpo: Está definido con el conjunto de marcos conceptuales contenido a su vez en un marco conceptual.

5.2. *Sintaxis de LMC*

La sintaxis de LMC está fundamentada en el bloque conceptual, este está constituido de: definiciones, interacciones, indagaciones, comprobaciones y elaboraciones. Ver figura 6.

Figura 6: Sintaxis LMC

Definición: como en todo lenguaje de programación, en LMC existe un bloque conceptual en el que se pueden hacer tres tipos de definiciones, una corresponde al enunciado, otra a las variables y una última a las anotaciones. Una definición de enunciado sirve para proponer o invocaciones o retornos. Las invocaciones son llamadas a marcos conceptuales ya constituidos, en tanto que el retorno es una definición explicita que regresa el control del marco conceptual hasta el punto en donde fue invocado. Una definición de variable sirve para constituir los contendores mínimos de la información esto en una declaración o para hacer uso de ellas estando ya definidas en una utilización. Por último se tienen las definiciones de anotación cuyo objetivo es el de documentar.

Interacción: por medio de las interacciones LMC permite al usuario gestionar entradas y salidas través de las cuales es posible comunicar condiciones deseadas para la ejecución de un programa.

Indagación: LMC propone un modelo basado en la formulación de preguntas en torno al estado que pueden tomar las definiciones, especialmente las variables, este tipo de indagación puede ser directo o recursivo. Se habla de una indagación directa aquella indagación que conduce a tomar un camino u otro una sola vez en tanto la indagación recursiva insiste en tomar un camino u otro un número veces definido para la recursión.

Comprobación: la comprobación permite definir escenarios en lo que los resultados par unas mismas condiciones pueden ser diferentes, dado el cambio incontrolado que en un momento dado pueden tomar las variables. Este concepto puede ser compilado como experimentación, contraste, demostración, argumentación, etc.

5.3. *Semántica de LMC*

La configuración de los marcos conceptuales de manera coherente constituye la semántica de LMC. Ver figura 7.

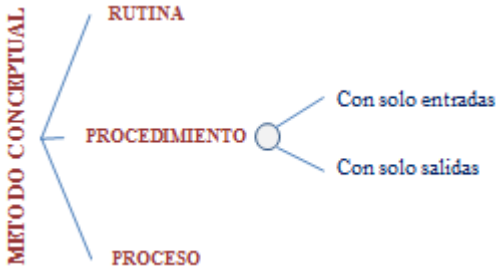

Figura 7: Semántica LMC

Rutina: esta define un conjunto de bloques conceptuales de manera coherente resolviendo un problema atómico, en el que no se recibe ni se expone información al ambiente, la rutina configura una solución completa dentro del modulo (Meyer, 1999).

Procedimiento: este define un conjunto de bloques conceptuales de manera coherente resolviendo un problema en el que se recibe o se produce información para el entorno de invocación.

Proceso: este define un conjunto de bloques conceptuales de manera coherente resolveindo un problema en el que se recibe y se produce información para el entorno de invocación.

La diferencia entre rutina procedimeitno y proceso radica fundamentalemente en el intercambio de información con el entorno.

6. Esquema Vs Diagrama

En LMC se producen esquemas, estos configuran los bloques de alto nivel del lenguajes. LMC propone dejar en segundo plano los conceptos de relaciones dejandolas implícitas a través de la secuencia y el anidamiento. Dado que el lenguaje busca armar sus abstracciones por la configuración de sus bloques tal y como se hace con un lego, las relaciones de composicion son simuladas por el secuenciameinto horizontal, las relaciones de herencia y realizacion son relaciones que se arman secuneciadas verticalmente entre métodos conceptuales. Ver figura 8.

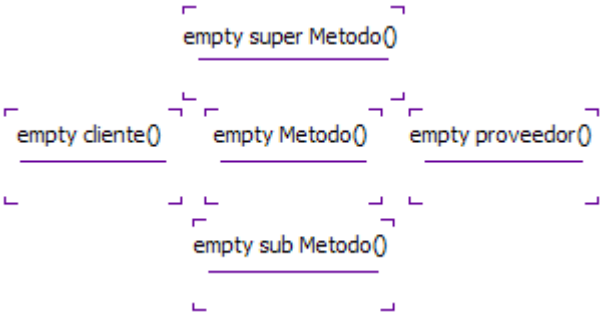

Figura 8: Relaciones implicitas para un bloque

Si el método tiene una categoria superior se ubica en la vertical arriba, si tiene una subclase se ubica abajo, sobre la horizontal el cliente se ubica a la izquierda en tanto que el proveedor se ubica aderecha. En la vertical el reuso es de caja blanca en

tanto que en la horizontal el reuso es de caja negra (Gamma et al., 1995).

6.1. Gestal en LMC

El esquema que se propone en LMC esta fundamentado en el cierre y la de agrupación, ver figura 9.

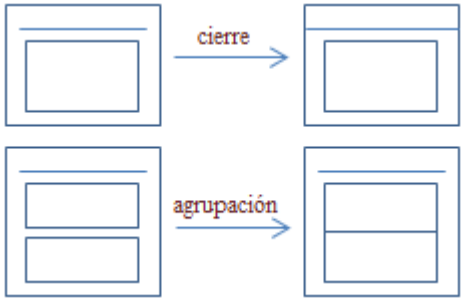

Figura 9: cierre y agrupación

El cierre propone un marco completo, cuyo propósito es afirmar la composición en el concepto que se embeba en el cuerpo del marco, en tanto que la agrupación afirma la cohesión del marco. Estas dos características visulamente son valiosas para manipular un marco y favorecer la coherencia y entendimiento. El esquema también esta diseñado para configurar la forma y el fondo. Ver figura 10.

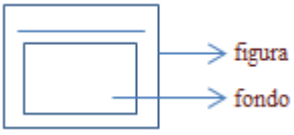

Figura 10: figura y fondo

6.2. Código de color en LMC

Los esquemas también manejan un código de color valioso para, resaltar aun mas su significado.

Para la definición se establece el color verde, este color propone confianza, tranquilidad y desarrollo (Heller, 2007). Ver figura 11.

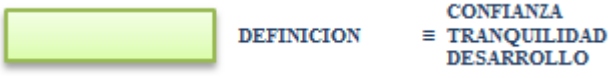

Figura 11: Color de la definición

Para la Interacción se establece el color Naranja, este color propone lo llamativo la transformación y la socialización (Heller, 2007). Ver figura 12.

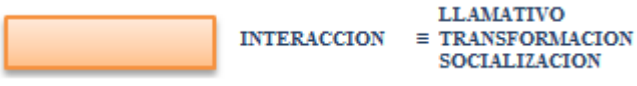

Figura 12: Color de la interacción

Para la indagación se establece el color azul, este color propone ciencia, lo ideal y lo funcional (Heller, 2007). Ver figura 13.

Figura 13: Color de la Indagación

Para la elaboración se establece el color rojo, este color propone, prohibición, peligro y dinamismo (Heller, 2007). Ver figura 14.

Figura 14: Color de la elaboración

Para la comprobación se establece el color amarillo, este color propone, iluminación, advertencia y creatividad (Heller, 2007). Ver figura 15.

Figura 15: Color de la comprobación

7. LMC Vs otros lenguajes

En LMC también se pueden expresar categorias semejante a clases presentes en los modelos orientados a objetos, por ejemplo si se tiene una clase en Java ver siguiente código.

```
public class X{
   private int a = 0;
   public void f(){/*codigo*/}
}
```

Este puede ser representado en UML (Booch et al., 2005a), (Booch et al., 2005b) como: ver figura 16.

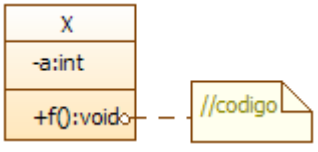

Figura 16: Clase en uml

Y en LMC se tiene el esquema, ver figura 17.

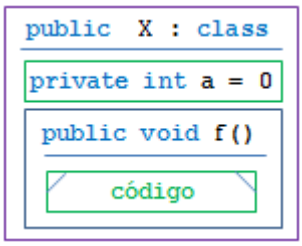

Figura 17: Categoria en LMC

En el lenguaje de representación según las producciones de LMC se tiene:

```
{
 public X:class |
 [
   { [ private int a = 0 ] }
   { public void f()|[{[codigo]}] }
 ]
}
```

8. Principios de LMC

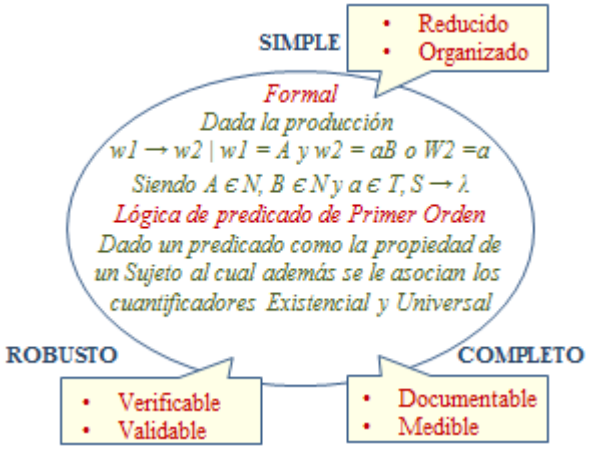

Figura 18: Pincipios de LMC

LMC es simple, robusto y completo; estos son los tres pincipios en los que se fundamenta el lenguaje. La simplicidad (Maeda, 2008), permite reducir el trabajo, gracias a un conjunto de abstracciones de mas alto nivel que las utilizadas en un lenguaje de programación, tan solo se manejan bloques, que se configuran, acorde al modelo que se desee representar; la organización de los bloques es automática, reduciendo la curva de aprendizaje; hay una marcada diferencia con los lenguajes de programación incluso con los de modelamiento, esta diferencia se basa en trabajar menos con texto en el caso de los lenguajes de programación así como remplazar el concepto nodo-arco en el caso de la diagramación, es decir el esquema LMC sintetiza texto y diagrama; para ambos casos, lenguaje programación y de diseño se reduce la complejidad (Morin, 2001). El esquema en LMC permite esbozar un concepto sin entrar en los detalles del lenguaje. LMC es robusto al permitir manejar de manera directa y transparente, conceptos que permiten verificar los algoritmos mediante la traza de sus fallos, y la validación de sus entradas y sus salidas; por otro lado es completo al permitir la documentación como mecanismo nativo y proveer métricas directas. LMC es formal por su estructura bien formada: vocabulario y reglas de producción bien definidas (Roman, 1990), (Iranzo, 2005). Ver figura 18.

9. Metricas de LMC

Basta con una breve reflexión par darnos cuenta que la métrica impregna la vida cotidiana, se mide con relojes, calendarios, notaciones musicales, chip de memoria, termómetros, índices de masa corporal, censos y muchos otros elementos que reducen el mundo a números y estadísticas (Robinson, 2007); de igual manera en el software es necesario contar con métricas que permitan tener un control sobre lo desarrollado. En LMC se proponen dos métricas, el balance de la incertidumbre y la densidad algoritmica (Bolaños et al., 2009).

9.1. Balance de la Incertidumbre

El desarrollo de software es un ejercicio creativo (Weinberg, 1971), en el que se parte de un dominio del problema con gran incertidumbre para llegar a un dominio de la solución con gran certidumbre. En el dominio del problema hay más preguntas que respuestas, mientras en el dominio de la solución hay más respuestas que preguntas, por lo menos con respecto al problema resuelto. Es de suponer que el ejercicio de desarrollar software esta enfocado en la eliminación paulatina de la incertidumbre para acercarse a la certidumbre permitida en la que se dé la solución. En este camino es de esperarse que se busquen mecanismos para facilitar el entendimiento del problema.

Aparecen dos constantes importantes las preguntas y las respuestas, un algoritmo generalmente está dotado de ambas, es importante entonces realizar las mejores preguntas que apunten a los orígenes del problema, cuyas respuestas necesariamente redunden en la solución. Como premisa fundamental de las preguntas se podría plantear:

- Todo problema por la inminencia de la incertidumbre produce preguntas.
- Si hay ausencia de preguntas por la naturaleza particular del problema, se tiene entonces un estado de certeza.

Por su parte las respuestas son el resultado de las preguntas o de los estados de certeza de un problema. Estas deben ofrecer la solución a un problema. Desde otro punto de vista se podría plantear que un problema se evidencia a través de las preguntas; mientras la solución se evidencia a través de las respuestas. Bajo esta consideración es de esperarse entonces que necesariamente, o por lo menos en su gran mayoría se produzca la dupla pregunta-respuesta. Desde un planteamiento deductivo podría pensarse en un marco de trabajo en donde no se vean problemas sino simplemente soluciones por tanto se eliminaría

el complejo mundo de las preguntas, sin embargo este mecanismo es mucho más complejo aunque no lejano del software.

Las respuestas están en el lado de la solución pero son originadas por preguntas o son en esencia un estado de verdad transitorio para un problema. Se puede plantear la siguiente premisa:

- Toda respuesta o proviene de una pregunta o de un estado transitorio de verdad

El desarrollo de un problema basado en un esquema lógico de programación sugiere que se realicen preguntas y respuestas, las cuales posibilitan que la transición de estados contenga la lógica necesaria para mostrar una solución. Ante estas evidencias vale la pena cuestionarse sobre la correlación que deben tener las preguntas y las repuestas, y producto de esta correlación podrían originarse los siguientes planteamientos.

1. ¿Una Pregunta debería originar una sola respuesta?
2. ¿Una pregunta debería originar varias respuestas?
3. ¿Varias preguntas deberían originar una sola respuesta?

Si se comienza por evaluar el segundo planteamiento se podría pensar, que el problema expresado en una única pregunta es resuelto a través de varios repuestas de las cuales alguna de ellas satisfacerá la pregunta, o simplemente de manera oculta de lo que se dispone como aparentes respuesta es de un protocolo en el que todas las aparentes repuestas hacen parte de un misma solución.

En el primer caso el problema puede ser resuelto con una de las respuestas en el segundo con la suma de todas las respuestas.

En el tercer planteamiento se estaría ante una situación en la que se podría pensar en una respuesta inteligente capaz de proveer la solución a varios cuestionamientos, o simplemente se estaría ante la evidencia de la redundancia de preguntas que conduce a la misma repuesta. En el primer caso en el que la respuesta es una conclusión inteligente que resuelve varios cuestionamientos se está ante un abstracción fuerte del problema en el que se debería descartar la redundancia; por otro lado puede existir redundancia de preguntas y ante lo que parecen varios cuestionamientos puede resultar más bien malos planteamientos.

El segundo y tercer planteamiento apuntan a un reflexión importante, ¿están en la muchas preguntas o muchas respuestas reflejado verdaderamente un problema de redundancia que debería ser normalizado para lograr un balance en el que las preguntas podrían corresponder uno a uno con las respuestas? Y no es porque se trate de eliminar la riqueza de la pregunta o de la respuesta; más bien de lo que se trata es de hacer converger cuestionamientos y soluciones. Esto es ventajoso en el manejo de la situación problemática. Algunos ejemplos, pueden ser:

1. a) ¿Qué numero resulta de 10/2?
   b) ¿Cuál es el número de dedos de una mano?

Las anteriores preguntas motivan una respuesta el número 5. Esta redundancia de preguntas podría centralizarse en una sola preguntas similar a: ¿Cuál es la mitad de los dedos de las manos? dentro de la cual se expresa las preguntas a y b.

2. ¿Cuál es el conjunto de 5 números naturales? respuesta que origina un conjunto infinito. Podría normalizarse al conjunto {1,2,3,4,5}.

En las dos consideraciones anteriores se puede ver el concepto pregunta respuesta, además como se puede generar la normalización de la repuesta o la pregunta; esta normalización que produce asignación uno a uno de preguntas y respuestas es lo que se denomina equilibrio del balance de la incertidumbre. Una buena forma de seguir el progreso del desarrollo de software es la de estimar el balance este puede ser de tres formas. Tal y como se ilustra en la figura 19.

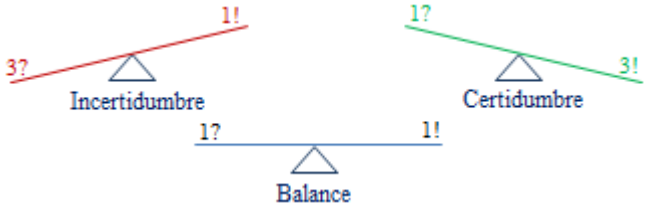

Figura 19: balance de la incertidumbre

El primer tipo de balance es el balance en el que pesan más las preguntas que las repuestas, este tipo de balance es de alta incertidumbre, existen más preguntas que respuestas y puede desencadenar difícil manejo en los desarrollos. El tipo de balance en donde las preguntas coinciden con las repuestas es el tipo de balance equilibrado, este resulta ser el más aconsejado como medio de atacar la incertidumbre, pues cada interrogante tiene su solución. El tercer tipo de balance tiene mayor peso en las repuestas, generalmente obedece a un conjunto de respuestas que hacen parte de protocolos o de respuestas múltiples. La tendencia en el software es a conseguir un balance equilibrado pues se tiene una correspondencia uno a uno del problema y la solución. Para lograr balancear equilibradamente el software es necesario eliminar la redundancia en las preguntas o en las respuestas con nuevas preguntas o nuevas respuestas mejor establecidas.

En LMC las preguntas corresponden a indagaciones en tanto las respuestas corresponden a definiciones, e interacciones; las elaboraciones por su parte pueden estar constituidas por preguntas y respuestas.

9.2. Densidad algorítmica

Este tipo de métrica es similar en la información al balance de la incertidumbre la diferencia radica en que indagaciones y definiciones se grafican como barras tratando de mostrar un gráfico que balanceado presenta el comportamiento de la campana de Gauss, ver figura 20.

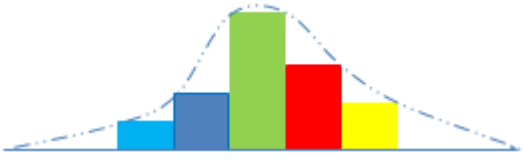

Figura 20: Densidad algorítmica

10. Plataforma coloso LMC

La plataforma coloso para LMC es una aplicación del lenguaje realizada en java, usando el framework SWT (Hatton, 2005). El software implementa una capa meta y una capa objeto. En la capa meta se conserva la esencia del lenguaje, mientras en la capa objeto se hace una aproximación a los lenguajes de programación imperativos. Ver figura 21.

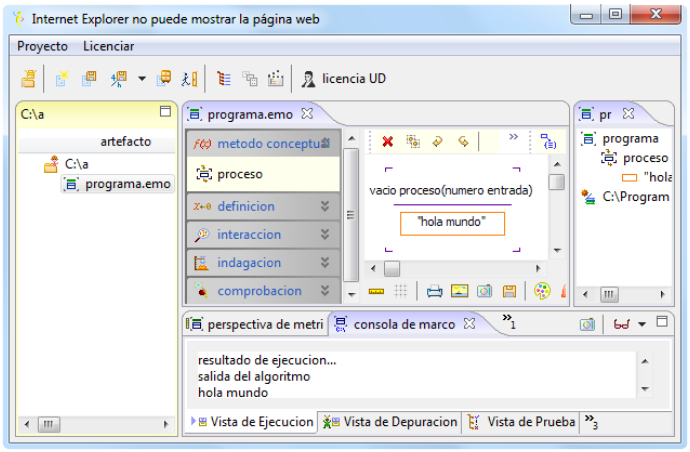

Figura 21: Software Coloso. `http://www.colosoft.com.co`

Todo marco conceptual es gestionado mediante diálogos emergentes en los que se sintetizan las posibilidades semánticas del lenguaje, esta primera capa de interacción con el usuario establece el primer filtro de las expresiones de LMC; el segundo filtro se hace una vez se corra el método conceptual; en segundo plano se lanza una aplicación al vuelo que se carga y compila en java, el resultado es subido y presentado en la primera capa sin que el usuario tenga que conocer los detalles del lenguaje base.

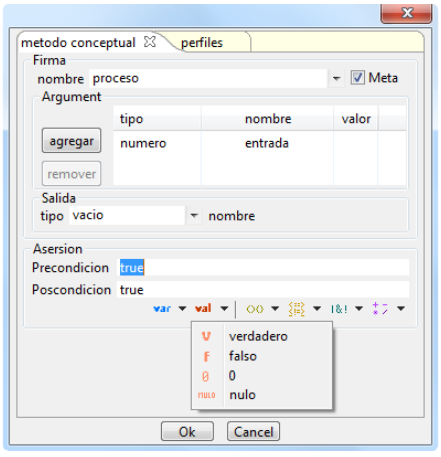

Figura 22: método conceptual

El método conceptual figura 22, puede gestionar entradas y salidas y las pre y pos condiciones, además todo marco sonceptual incorpora una barra de herramientas en la que se encuentran las variables que pueden ser utilizadas al momemnto de usar un marco determinado, también se presentan los valores sugeridos, operadores y agrupadores útiles en la construcción de sentencias, la barra de herramientas se activa acorde a las condiciones de validez de la instrucción en construcción.

Un marco puede ser medido, ejecutado, depurado, validado y verificado; las salidas pueden ser evidenciadas en cada una de las vistas dedicadas para cada propósito.

La depuración permite hacer el seguimiento paso a paso del método conceptual, este seguimiento puede ser visualizado en el marco de diseño, en el árbol de presentación, y en la vista de depuración, en esta última puede evidenciarse el estado actualizado de las variables definidas en el marco, al punto de seguimiento en el que se encuentre el programa. Ver figura 23.

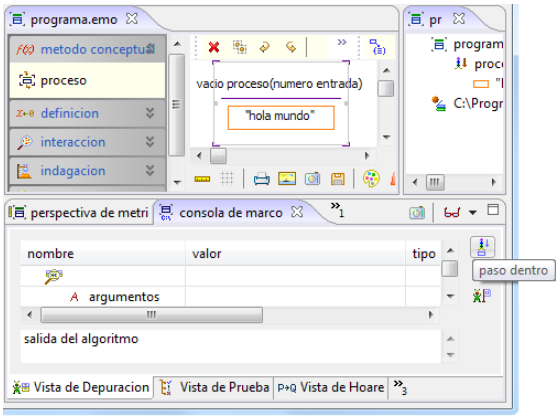

Figura 23: vista de depuración

La verificación y validación de los métodos conceptuales pueden evidenciarse en sus respectivas vistas, la verificación se visualiza como un árbol en el que se mapean errores, fallos y defectos. En la perspectiva de validación se tabula la tripleta de Hoare (Hoare, 1969), Hoare (2009) resaltando la evaluación correspondiente al método. Figura 24. Ambos marcos se basan en un framework de prueba implementado para LMC.

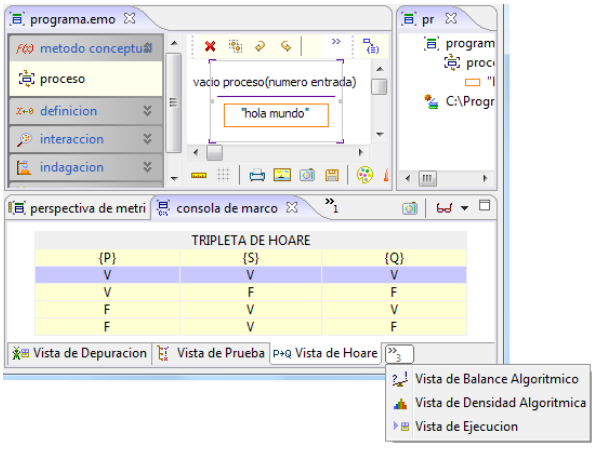

Figura 24: vista de validación

LMC presenta una vista de métrica en el que se enumeran el cojunto de marcos utilizados y posibilita junto con la vista de densidad algorítmica y balance algorítmico, tener una visión de la complejidad y la tendencia que caracteriza un determinado

método conceptual, se proponen un conjunto de recomendaciones basadas en la complejidad ciclomática (McCabe, 1976) y el número mágico 7±2 (Miller, 1956), ver figura 25.

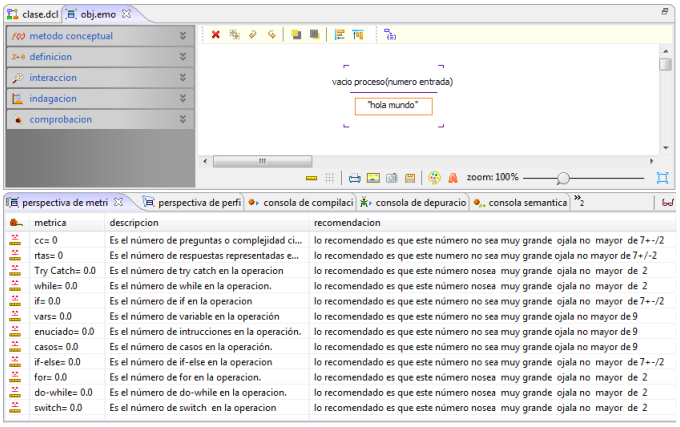

Figura 25: vista de métrica

Otra de las características que provee LMC, es la facilidad de integración con lenguajes como UML, este puente estrecha la brecha entre el diagrama de clases y el lenguaje de programación, específicamente esta facilidad permite la generación del código de una operación perteneciente a una clase, y el algoritmo resuelto en un método conceptual y que esta asociado a esa operación. Figura 26.

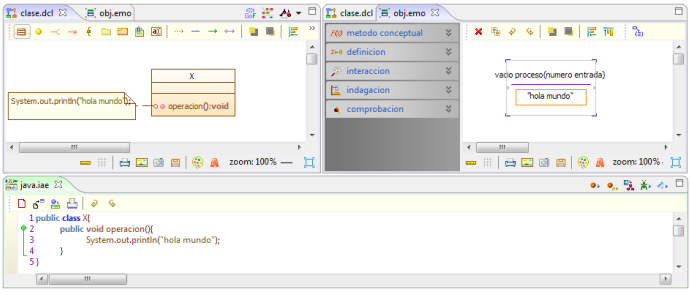

Figura 26: integración uml/cfl/java

11. Casos de Estudio

A continuación se describen las pruebas realizadas para validar los principios simplicidad, robustez y completitud de LMC.

La característica de simplicidad la asociamos con la variable tiempo empleado en el desarrollo de un determinado problema, para ello realizamos la comparación del esfuerzo, medido en tiempo necesario para resolver el conjunto de algoritmos que conforman un curso de programación y algoritmos. La prueba incluyó la realización de un total de 20 algoritmos y fue aplicada a los profesores que imparten el área. El curso normalmente se dirige con herramientas como DFD (Cárdenas et al., Citado 2012), (PSeInt, Citado 2012) o lenguajes de programación como java; en la muestra de 20 problemas que conforman el núcleo del curso se cuentan problemas de: recorridos, intercambios, búsquedas y ordenamientos. Se distribuyeron 5 problemas por cada tema y se conformaron dos grupos “dos profesores por grupo”, un grupo resolvió los algoritmos usando la herramienta con que venían trabajando “DFD”, a este grupo lo llamaremos alterno; el segundo grupo trabajó con LMC, “previo entrenamiento e ilustración del lenguaje”, a este grupo lo llamaremos LMC. En la tabla 1. se tabulan los tiempos empleados en minutos, junto con el resultado obtenido del promedio ecuacion 2. y desviación estándar ecuacion 3.

$$\mu = \frac{1}{n}\sum_{i=1}^{n} a_i = \frac{a_1 + a_2 + ...+_n}{n} \tag{2}$$

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(X_i - \mu)^2}{n}} \tag{3}$$

Tabla 1: Prueba de simplicidad

Tema	Grupo Alterno	Grupo LMC
intercambio	22	20
recorridos	30	23
búsquedas	40	27
ordenamientos	80	30
μ	**43**	**25**
σ	**22.29**	**3.80**

El promedio del tiempo de desarrollo de los algoritmos en LMC fue muy inferior al grupo alterno. La dispersión de la muestra para LMC fue baja en tanto en el grupo alterno fue alta, la razón fundamental se vio reflejada en un algoritmo en particular de ordenamiento, el problema de las torres de hanoi (Pickover, 2011), este problema fue resuelto fácilmente en LMC, el grupo alterno se vio limitado a resolverlo de forma iterativa dado que la recursión no puede ser implementar directamente en DFD, esto aumentó considerablemente el tiempo de solución.

La siguiente prueba se orientó a establecer la robustez, en este punto se abrió la posibilidad de utilizar un lenguaje de programación si se deseaba. La prueba consistió en romper los algoritmos con entradas no válidas, para lo cual se dio un espacio de una hora. En la tabla 2. se presenta el número de algoritmos probados (exitosamente) por categoría.

Tabla 2: Prueba de Robustez

Tema	Grupo Alterno	Grupo LMC
intercambio	5	5
recorridos	3	5
búsquedas	2	5
ordenamientos	1	5
μ	**2.75**	**5**
σ	**1.47**	**0**

LMC brindó gran facilidad por el manejo directo y transparente que hace del concepto de aserción. Para el grupo alterno, que tomó la opción de cambiarse a java se convirtió en una tarea dispendiosa dado que tuvieron que migrar los algoritmos, no se manejaba el concepto *assert* y redujo la tarea a la realización de comparaciones. En esta prueba se acentúa la diferencia entre

LMC y el grupo alterno. En el caso de LMC a pesar de brindar la posibilidad de tomar un código generado java no se utilizo dada la facilidad proveída por el lenguaje para la realización de aserciones "pre y poscondiciones".

La tercera prueba fue dirigida probar condiciones de completitud, para ello se pidió a los grupos que explicasen el modelo o código que resolvía cada problema, valiéndose además de alguna métrica sobre los algoritmos. El tiempo dado para esta actividad fue de una hora. En la tabla 3. se presenta el número de algoritmos documentados.

Tabla 3: Prueba de Completitud

Tema	Grupo Alterno	Grupo LMC
intercambio	5	5
recorridos	5	5
búsquedas	5	5
ordenamientos	5	5
μ	**5**	**5**
σ	**0**	**0**

Todos los algoritmos para ambos grupos fueron documentados sin embargo esta documentación en el grupo alterno fue realizada aparte, esta decisión fue tomada dado que se había empleado hasta el momento dos herramientas diferentes e implicaría dos documentos separados, sino se hacia una documentación en una tercera herramienta; además esta documentación no fue enriquecida con criterios de métricas; en tanto la documentación del grupo LMC fue integrada en el programa, apoyándose en las métricas de generación automáticas que propone el lenguaje.

Las anteriores pruebas dan punto de partida favorable a LMC; sin embargo, se proponen un grupo de pruebas a futuro mucho más extensivas, en donde se hagan comparaciones con lenguajes de programación, y otras herramientas que faciliten el modelado y solución de algoritmos. Además se proponen pruebas para determinar la curva de aprendizaje sobre grupos de estudiantes. También se planean pruebas en un ambiente empresariales dada la trazabilidad e integración que facilita le lenguaje con lenguajes de programación como java y lenguajes de diseño como UML.

12. Conclusiones

Las pruebas alfa realizadas sobre LMC, muestran como el lenguaje facilita la resolución de problemas computacionales, por el grado de abstracción que provee, ocultando gran parte de la complejidad asociada al conocimiento de una semántica y sintaxis propias de un lenguaje computacional.

Los esquemas utilizados por LMC, como forma de representación gráfica, eliminan la complejidad que introducen la relaciones en un modelo gráfico convencional, en lugar de ello se emplea el ensamble acudiendo al modelo mental de secuencia y anidamiento.

La hipótesis debe ser contrastada con un amplio espectro de usuarios a los que se les debe instruir en el lenguaje, con el cuidado de mostrarlo apartándose de esquemas conocidos como por ejemplo el modelo O.O, el modelo estructurado, entre otros, el objetivo deber ser dirigido a presentar los mecanismos de abstracción y contextualización propios del lenguaje.

LMC enfatiza en la resolución de problemas acudiendo a un vocabulario más científico, en el que predominan la indagación (observación), la definición de variables, la definición de pruebas y experimentos y las elaboraciones; las características computacionales propias de los lenguajes de programación quedan en segundo plano.

13. Trabajos futuros

Dentro de los trabajos que se proponen para continuar desarrollando el lenguaje están:

- Trazabilidad a otros modelos: al momento se ha implementado solamente la interacción con UML y la trazabilidad al lenguaje de programación java, se propone también otras trazabilidades a lenguajes de programación entre los que se cuentan: c++, c#, php, y python.
- Desarrollo de métricas: las métricas propuestas acerca del balance y densidad algorítmica sugieren un control en la abstracción de un método conceptual materializado con un número que se asocia a una condición cognitiva según la teoría propuesta en el número mágico. Se plantea desarrollar otras métricas basadas en redes bayesianas (Fenton and Neil, 1999), (Fenton et al., 2007).
- Desarrollo de patrones: entre los aportes más significativos en ingeniería de software se cuentan los patrones; estos se han propuesto desde el análisis (Fowler, 1997), la arquitectura (Shaw and David, 1996), (Buschmann et al., 2002) el diseño (Gamma et al., 1995), (Freeman et al., 2004), (Larman, 2003) incluso los procesos de software (Bolaños et al., 2011). En LMC se plantea estudiar diferentes posibilidades de patrones y proponerlos como mecanismos de ayuda en la utilización del lenguaje.
- Desarrollo de libretos: los libretos son otro mecanismo dirigido a resolver "problemas típicos"a diferencia de los patrones los libretos son aplicaciones mas precisas al cumplimiento de un objetivo (Medina et al., 2012), con libretos es posible construir librerías con soluciones ya listas de un determinado problema.
- Refinamiento de la capa meta: este trabajo propone robustecer el esquema meta dirigido a solucionar problemas en lógicas: multivalente, modal, temporal, intuicionista, y no monótona (Iranzo, 2005).

English Summary

Conceptual Frameworks Language –CFL– Nearing Design and Implementation

Abstract

This paper presents the Conceptual Frameworks Language CFL, it aims to bridge the gap between programming languages and design languages, using the mechanism of schematizing, this proposed change the complexity of the syntax of programming languages and complexity of the diagramming for ease of assembly and nesting of frames or conceptual blocks like building brick, we present the possibilities offered by CFL as a Language nearest to solving problems using computational and scientific vocabulary, which is transparent to the user, CFL raises comparisons and integrations with languages like java and UML, we propose metrics and develop the platform in java language.

Keywords:

Language, scheme, metrics, contextualization, abstraction, vocabulary, syntax, semantics.

Agradecimientos

Este proyecto de investigación ha sido desarrollado gracias al soporte académico del grupo de investigación GICOGE - Grupo Internacional de Investigación en Informática Comunicaciones y Gestión del Conocimiento-, A las universidades Distrital de Colombia y Pontificia de Salamanca de España por el acompañamiento y apoyo económico.

Referencias

Bolaños, S., González, R., Medina, V., 2011. Patterns of software development process. International Journal of Interactive Multimedia and Artificial Intelligence 1 (4).

Bolaños, S., Medina, V., Joyanes, L., 2009. Principios para la fromalización de la ingeniería de software. Ingeniería 14, 31–37.

Booch, G., Rumbaugh, J., Jacobson, I., 2005a. Addison-Wesley.

Booch, G., Rumbaugh, J., Jacobson, I., 2005b. Addison-Wesley.

Budd, T., 1994. Addison-Wesley.

Budge, E. A., Wallis, E. S., 1857-1934. London : British Museum.

Buschmann, F., Meunier, R., Rohner, H., Sommerland, P., Stal, M., 2002. Jonh Wiley.

Chomsky, N., 2004. Siglo ventiuno editores.

Cárdenas, F., Daza, E., Castillo, N., Citado 2012. Dfd.
URL: `http://freedfd.googlecode.com/files/FreeDFD-1.1.zip`

Fenton, N., Neil, M., 1999. Software metrics: successes, failures and new directions. Journal of Systems and Software 47 (2).

Fenton, N., Neil, M., Marsh, W., Hearty, P., Marquez, D., Krause, P., Mishra, R., 2007. Predicting software defects in varying development lifecycles using bayesian nets. Information and Software Technology 49 (1).

Fowler, M., 1997. Addison Wesley.

Freeman, E., Freeman, E., Sierra, K., Bates, B., 2004. O'Reilly.

Gamma, E., Helm, R., Johnson, R., Vlisides, J., 1995. Addison-Wesley.

Hatton, R., 2005. O'Reilly & Associates.

Heller, E., 2007. Editorial Gustavo Gili.

Hoare, C., 1969. An axiomatic basis for computer programming. Communication of the ACM 12 (10).

Hoare, C., 2009. Viewpoint. retrospective: an axiomatic basis for computer programming. Communication of the ACM 52 (10).

Iranzo, P., 2005. Alfaomega Ra-ma.

Kuhn, T., 1971. Fondo de Cultura Económica.

Larman, C., 2003. Prentice Hall.

Maeda, J., 2008. Gedisa S.A.

McCabe, T., 1976. A complexity measure. IEEE Journal & Magazines SE-2 (4).

Medina, V., Bolaños, S., González, R., 2012. Process management software as a script, and the script as a pattern. IACSIT 1 (2).

Meyer, B., 1999. Prentice Hall.

Miller, G. A., 1956. The magical number seven, plus or minus two: Some limits on our capacity for processing information. Psychological Review 63 (2).

Morin, E., 2001. Gedisa S.A.

Pickover, C., 2011. Sterling Publishing Co. Inc.

PSeInt, Citado 2012. Pseint.
URL: `Online:http://pseint.sourceforge.net/`

Robinson, A., 2007. Edisiones Paidós.

Roman, S. S., 1990. Díaz de Santos.

Rosen, K., 2004. Mc Graw Hill.

Shaw, M., David, G., 1996. Prentice Hall.

Teufel, B., Schimidt, S., 1995. Addison-Wesley.

Thorpe, S., 2000. Norma S.A.

Tucker, A., Noonan, R., 2002. McGraw Hill.

Weinberg, G., 1971. Van Nostrand Reinhold.