Foundations for a development methodology software agile model – driven from a modeling graphic customer-oriented language

Duque M. Nathalia B., Pulgarín M. Mauricio A. y Ríos P. Jorge I.
Maestría en Ingeniería de Sistemas y Computación
Universidad Tecnológica de Pereira
Pereira, Colombia

***Abstract*— The software modeling languages have been designed for systems engineers, computer or software developers. This is why they are in charge of interpret customer's specifications and build the business model that after a process of software engineering it will become into software. In this point it begins to open the gap between customer specifications and software development. This work provides an alternative to reduce the semantic loss during communication between the client and the developer, using business rules that can be modeled by the customer from a graphical modeling language that is easy to learn and using the principles of the development of model-driven software so that by automating the coding process it can be ensured that the specifications are modeled by the customer and that they are in the developed software.**

***Keywords*—Software Engineering, Software Development, Ágile Methodologies, Model Driven Development, Graphic Model Languages**

I. INTRODUCCIÓN

A Partir de la crisis del software en la década del 70, los esfuerzos para hacer más eficientes los procesos de desarrollo de software se centraron fundamentalmente en un actor: los desarrolladores. El cliente participaba durante la elicitación de requerimientos pero era ignorado hasta la etapa de pruebas, provocando normalmente que las aplicaciones no cumplieran sus expectativas [1].

Como alternativa frente a esta situación surgen en el año 2001, con el Manifiesto Ágil [2] las denominadas metodologías ágiles y posteriormente otros trabajos como [1] y [3], que proponen contar con la colaboración del cliente durante todo el proceso de desarrollo, permitiendo la retroalimentación continua.

El desarrollo de software debe estar completamente orientado a los objetivos de las organizaciones [4], de allí que el proceso de modelado deba enfocarse en ello, pero las metodologías de desarrollo de software tradicionales trabajan desde una perspectiva tecnológica lo cual dificulta que los expertos del negocio puedan tener un alto grado de participación en el modelado de las soluciones que requieren. Los lenguajes de modelado han sido ideados para ingenieros de sistemas, informáticos o desarrolladores de software, lo cual explica que para alcanzar un buen dominio de ellos se deben tener conocimientos previos y hacer un curso especializado de modelado, en consecuencia, el cliente, quien es el experto que mejor conoce el negocio, no es quien lo modela y por esta razón existe una brecha entre las especificaciones del cliente y el desarrollo de software [3].

Nuestra propuesta consiste en involucrar al cliente en el proceso de desarrollo desde el inicio, tal y como sucede en las metodologías ágiles [5], pero de forma tal que sea él quien modele las reglas para las funcionalidades genéricas [6] que desea realizar a través del software, utilizando un lenguaje de modelado gráfico que sea fácil de aprender y que genere un canal de comunicación asertivo entre cliente y desarrollador. Para ello definimos un lenguaje de modelado gráfico llamado LMR (Lenguaje de Modelado Real), una Gramática Libre de Contexto (GLC) que permite validar el modelo construido y un intérprete que siguiendo los principios del Desarrollo Dirigido por Modelos (MDD) y utilizando la GLC interpreta y valida el modelo para generar la aplicación.

II. LENGUAJE DE MODELADO GRÁFICO LMR

En [6] se ha establecido que en todo proyecto de software, un alto porcentaje de sus funcionalidades presentan similar comportamiento, como son: "Alta", "Baja", "Modificación" y "Consulta" (ABMC). Esto hace posible plantear una solución que permita independizarse del elemento particular, y generalice el comportamiento descrito para cualquier entidad. Tomando [6] como punto de partida, se definieron las siguientes operaciones genéricas para todo sistema de información: actualizar, procesar, buscar y leer, buscar y procesar, insertar, borrar, reportar, extraer información, enviar información

A partir de estas funcionalidades genéricas, se realizó la clasificación de los elementos de un sistema de información como Elementos Estructurales y Elementos Operacionales

978-1-4799-6717-9/14/$31.00 ©2014 IEEE

según se muestra en las tablas I y II.

Para cada uno de los elementos estructurales y operacionales se estableció una representación gráfica en LMR teniendo en cuenta las características deseables para los lenguajes de modelado gráficos enunciadas en [7] y [8].

Posteriormente se establecieron las reglas que permiten determinar la semántica del modelo y su correspondiente metamodelo.

TABLA I
ELEMENTOS ESTRUCTURALES DE UN SISTEMA DE INFORMACIÓN

Elemento Estructural	Definición
Entidad	Representación abstracta de una cosa u objeto del mundo real, con existencia independiente.
Atributo	Representación abstracta de la descripción individual de una o varias características de las entidades.
Restricción real	Representación abstracta de la limitación o reducción real impuesta a un componente del sistema.
Entidad compuesta	Representación abstracta de una cosa u objeto del mundo real que se forma a partir de entidades reales.
Regla	Representación abstracta de una acción a ejecutar sobre un elemento del sistema, se utiliza para inicializar un componente o validarlo.
Relación	Representación abstracta del vínculo entre dos elementos del sistema.
Plantilla	Representación abstracta de un formato para mostrar información del sistema.

TABLA II
ELEMENTOS OPERACIONALES DE UN SISTEMA DE INFORMACIÓN

Elemento Operacional	Definición
Buscar y presentar	Representación abstracta de la consulta y muestra de información.
Buscar y procesar	Representación abstracta de la consulta y manipulación de información.
CRUD	Representación abstracta de la creación, obtención, actualización y borrado de información sin llevar a cabo un proceso.
Procesar	Representación abstracta de la transformación de información a partir de parámetros de entrada.
Reportar	Representación abstracta de la presentación de la información a partir de parámetros y condiciones de entrada.
Extraer	Representación abstracta de la obtención de información a partir de parámetros de entrada.
Enviar	Representación abstracta de la transmisión de información en el sistema.

TABLA III
REGLAS DE VALIDACIÓN PARA LA SEMÁNTICA DEL MODELO

REGLAS
1. Un atributo puede pertenecer a varias entidades.
2. A los atributos que son primarios se les antepone una llave y después del nombre se ponen las letras PK.
3. Las relaciones entre atributos y entidades son de agregación.
4. Las relaciones entre atributos y reglas son de ejecución.
5. Las restricciones se ubican en medio de las relaciones
6. Las relaciones entre entidades y entidades compuestas son de agregación.
7. Las relaciones que llegan a los componentes operacionales son de agregación.
8. Las relaciones que salen de los componente operacionales son de ejecución.

III. Representación gráfica de los procesos estandarizados de un sistema de información en LMR

Una vez definidas las características de LMR se procede a establecer la representación gráfica de cada uno de los elementos operacionales y estructurales, como se presenta en Fig. 1. – 5.

Fig. 1. Representación en LMR para el elemento Entidad

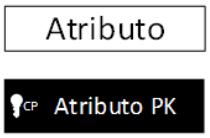

Fig. 2. Representación en LMR para el elemento Atributo

Fig. 3. Representación en LMR para el elemento Restricción

Fig. 4. Representación en LMR para el elemento Entidad Compuesta

Fig. 5. Representación en LMR para el elemento Regla

Fig. 6. Representación en LMR para el elemento Relación

Fig. 7. Representación en LMR para el elemento Plantilla

Fig. 8. Representación en LMR para el elemento Buscar y Presentar

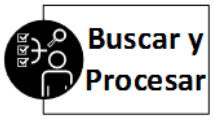

Fig. 9. Representación en LMR para el elemento Buscar y Procesar

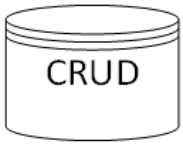

Fig. 10. Representación en LMR para el elemento CRUD

Fig. 11. Representación en LMR para el elemento Procesar

Fig. 12. Representación en LMR para el elemento Reportar

Fig. 13. Representación en LMR para el elemento Extraer

Fig. 14. Representación en LMR para el elemento Enviar

IV. GRAMÁTICA LIBRE DE CONTEXTO PARA LMR

A partir de las reglas enunciadas para validar la semántica del modelo que se observan en la tabla III y utilizando los conceptos de gramaticalidad y creatividad de [9] se creó una GLC definiendo un subconjunto de todas las secuencias (finitas o no), que se puedan formar mediante la concatenación de los elementos de un alfabeto finito de símbolos terminales. De acuerdo con [9] los símbolos del alfabeto son los símbolos que realmente aparecen en el modelo y nunca aparecerán del lado izquierdo de una producción. La gramática LMR se define formalmente mediante la cuaterna (1).

$$G = \{\Sigma, N, S, P\} \tag{1}$$

En la cuaterna definida, Σ es el conjunto finito de los símbolos terminales que representan cada uno de los elementos del LMR como se muestra en la Tabla IV; N es un conjunto finito de símbolos no terminales que de acuerdo con [9] son los metasímbolos que deben ser definidos por otras producciones y se presentan en la Tabla V; S es un símbolo terminal básico, axiomático según la definición de [9] que describe las oraciones enteras de un lenguaje natural; P es un conjunto finito de reglas que indica cómo se pueden generar los modelos y es un subconjunto de (2).

TABLA IV
SÍMBOLOS TERMINALES DEL ALFABETO DEFINIDO PARA LA GRAMÁTICA LMR

Elemento Terminal	Símbolo Terminal
Entidad	ent
Entidad compuesta	entc
Atributo	atr
Restricción	rest
Regla de inicialización	regi
Buscar y presentar	bp
Buscar y procesar	bpr
CRUD	crd
Reportar	rp
Enviar	env
Extraer	ex

$$P \subseteq (N \cup \Sigma *)N\,(N \cup \Sigma *)\ \ (N \cup \Sigma *) \tag{2}$$

A partir de (2) se creó el conjunto de reglas gramaticales presentado en la Tabla V, tal que (3), lo cual permite que LMR sea 100% computable por medio de la creación de un autómata capaz de interpretar cualquier modelo entero (representado como M) creado con el lenguaje.

$$P = \{R1, R2, \ldots, R17\} \tag{3}$$

Para cada una de las reglas gramaticales se definió el metamodelo correspondiente, tal como se presenta Fig. 15. – 22.

TABLA V
SÍMBOLOS NO TERMINALES DEL ALFABETO PARA LA GRAMÁTICA LMR

Elemento No Terminal	Símbolo No Terminal
PBP	Proceso de buscar y procesar
PBPR	Proceso de buscar y presentar
PCRD	Proceso de hacer CRUD
PPR	Proceso de hacer un proceso
PRP	Proceso de reportar
PEX	Proceso de extraer información
PENV	Proceso de enviar información
ENT	Entidades
CENT	Conjunto de entidades
ATR	Atributos
CATR	Conjunto de atributos
CREG	Conjunto de reglas
COND	Condicionalidad
CONDPR	Condicionalidad de procesar

TABLA VI
REGLAS GRAMATICALES PARA LMR

R1	M ⟶ PBP|PBPR|PCRD|PPR|PRP| PEX | PBP M|PBP M |PCRD M |PPR M |PRP M | PEX M |PENV M
R2	PBP ⟶ CATR bp ENTBP
R3	ENTBP ⟶ ENT | ENT M
R4	ENT ⟶ ent | ent CREST | ent CENT | entc | entc CREST | entc CENT
R5	CENT ⟶ ent | ent ENT | ent CENT | entc | entc ENT | entc CENT
R6	CATR ⟶ ATR | ATR CATR
R7	ATR ⟶ atr | atr CREG
R8	CREG ⟶ reg i | reg v |reg i reg v
R9	PBPR ⟶ CATR bpr ent COND | CATR bpr entc COND
R10	PCRD ⟶ crd ENT
R11	PPR ⟶ CATR pr | CATR pr CONDPR
R12	CONDPR ⟶ CENT COND | COND
R13	COND ⟶ CREG |PENV | CREG PENV | ε
R14	PRP ⟶ rp CENT
R15	PENV ⟶ env CENT
R16	PEX ⟶ ex CENT
R17	CREST ⟶ rest | rest CREST

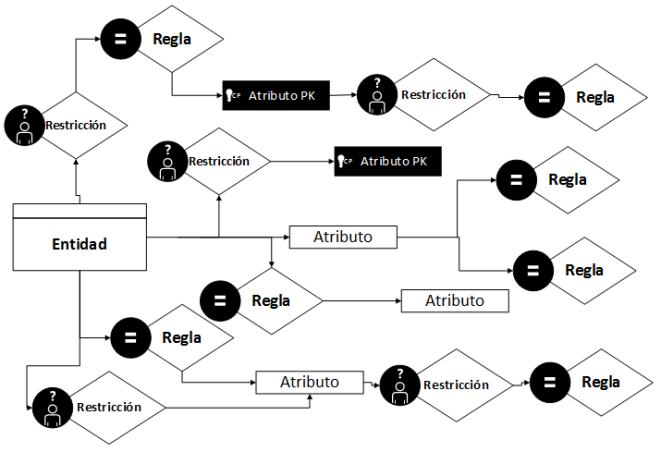

Fig. 25. Metamodelo de una Entidad bien formada

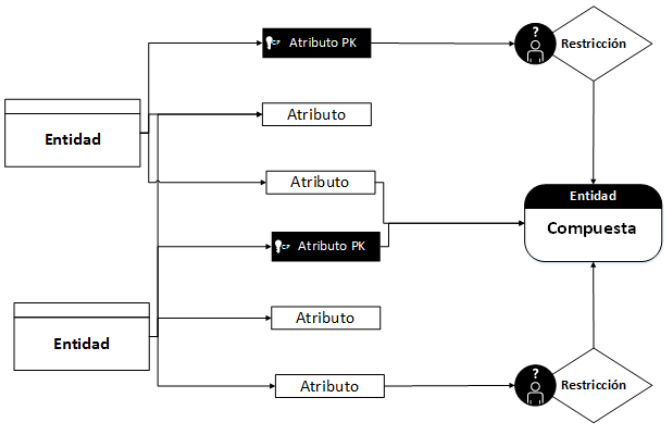

Fig. 16. Metamodelo de una Entidad Compuesta bien formada

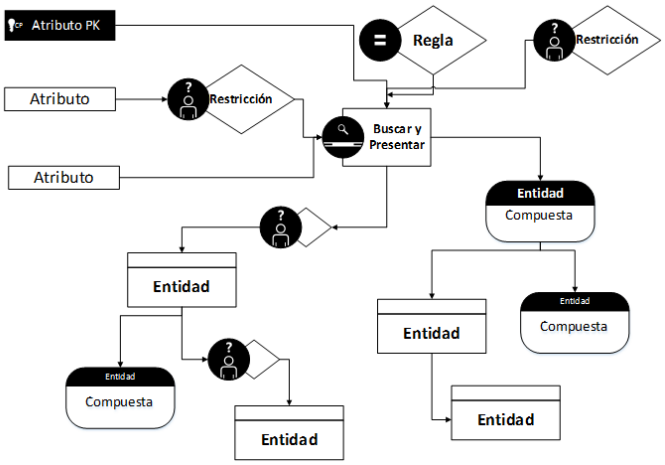

Fig. 17. Metamodelo proceso buscar y presentar bien formado

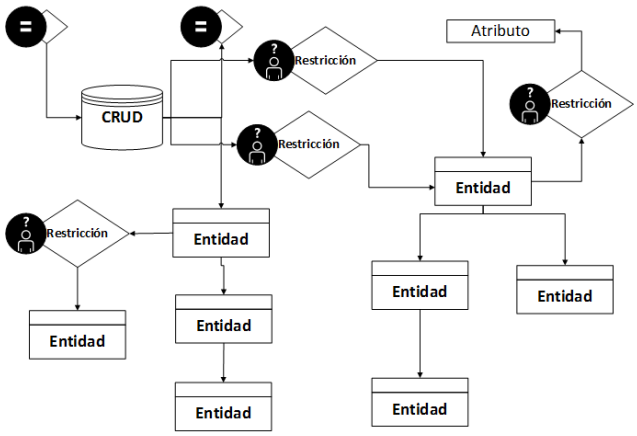

Fig. 18. Metamodelo proceso CRUD bien formado

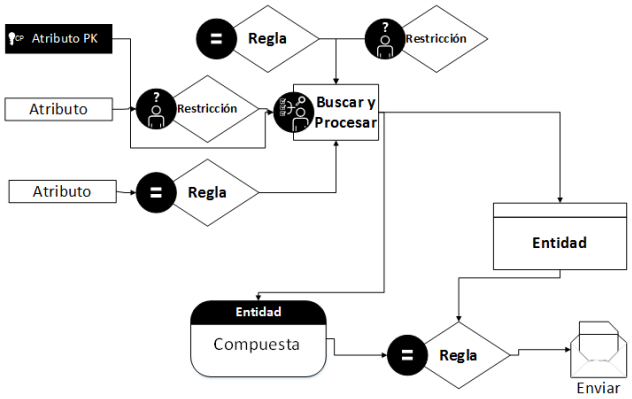

Fig. 19. Metamodelo proceso buscar y procesar bien formado

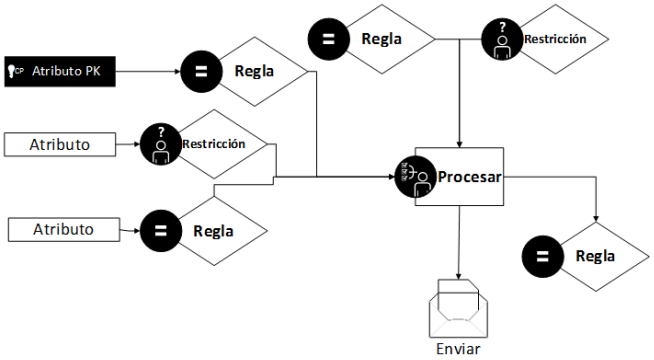

Fig. 20. Metamodelo acción de procesar bien formada

Teniendo definido el metamodelo para cada una de las reglas obtenidas a partir de la GLC, es posible verificar que los modelos del negocio se encuentren bien construidos. El paso siguiente en la metodología propuesta con este trabajo consiste en traducir el modelo construido en LMR a código XML (Extensible Markup Language) para que sea interpretado por un intérprete de GUI (Graphic User Interface) que genera la aplicación final. La intérprete es la herramienta por medio de la cual se busca garantizar que la solución generada sea conforme al modelo elaborado por el cliente.

V. Resultados Obtenidos

La ejecución de este trabajo arroja como resultado la formulación del lenguaje de modelado gráfico LMR, el cual tiene como cualidad diferenciadora frente a otros lenguajes de la misma naturaleza, el poder ser utilizado directamente por el cliente.

Se obtuvo una GLC a partir de la cual se validan los modelos construidos en LMR. Se construyó el intérprete para código XML que valida el modelo y genera la aplicación final.

Con el trabajo presentado en este artículo, se obtuvieron los casos de desarrollo de proyectos de software exitosos [10] y [11]. En las Fig. 7 y Fig. 8 se observan dos de los modelos construidos en LMR para [10].

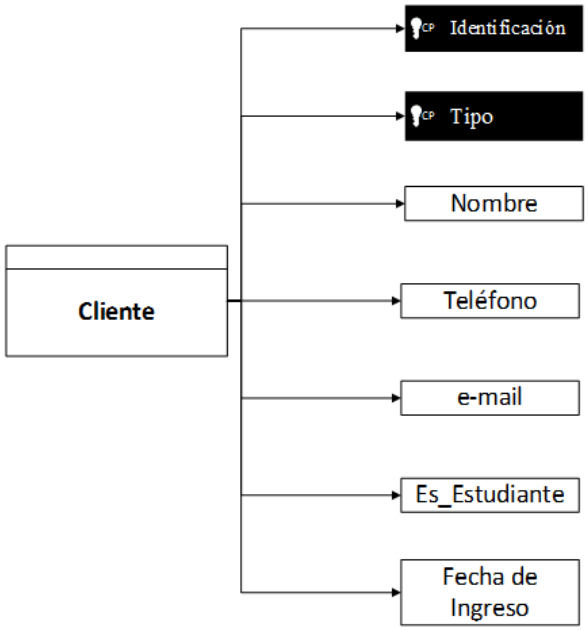

Fig.21. Modelo entidad cliente [10]

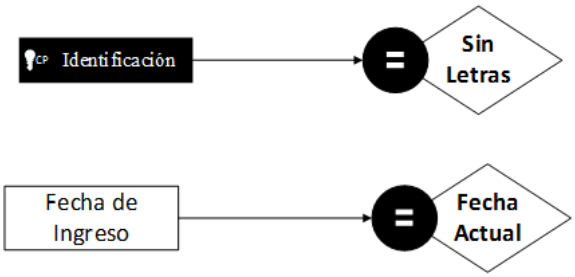

Fig. 22. Modelo de reglas para los atributos identificación y fecha de ingreso [10]

La interfaz gráfica generada con el intérprete de LMR para [10] puede observarse en Fig. 23.

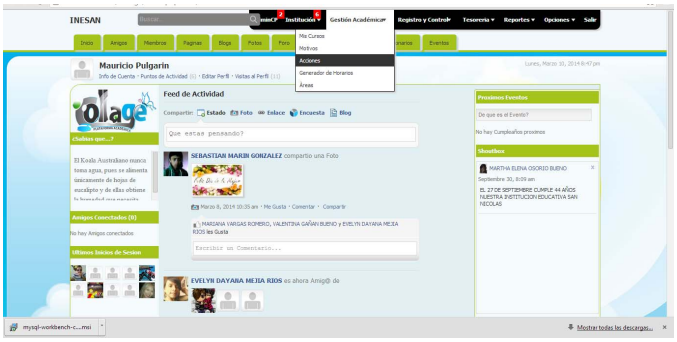

Fig. 23. GUI Software para administración de instituciones educativas[10]

Para el proceso de desarrollo de [10], se contó con la participación del Rector de la Institución Educativa, quien se constituyó como cliente. El desarrollo inició con la reunión del equipo de desarrollo con el cliente. Se explicó al rector y a su equipo de colaborares cuál era la metodología a utilizar. Posteriormente, se realizó una sesión de explicación de

modelado en LMR y a partir de allí se realizó el modelado de las funcionalidades que tendría el producto. Se elaboraron los modelos en papel para cada uno de los elementos de la solución y posteriormente se codificaron en XML para que de esta forma fueran interpretados, obteniendo así la interfaz gráfica de usuario de la aplicación.

VI. Conclusiones y Trabajo Futuro

El trabajo presentado aporta las bases de una nueva alternativa metodológica para desarrollo de software buscando aumentar la productividad y promoviendo un canal de comunicación entre el cliente y los desarrolladores, que permite disminuir la pérdida de semántica generada al utilizar los enfoques tradicionales, durante la elicitación de requerimientos.

Si bien, existen propuestas como [1] y [3] a través de las cuales se involucra al cliente dentro del proceso de desarrollo de software utilizando el enfoque colaborativo, la interpretación de las especificaciones sigue estando en manos de los desarrolladores. Utilizando LMR, el modelado de las especificaciones parte del cliente y no de una interpretación que puede ser errónea y con ello generar complicaciones adicionales a las inherentes a cualquier proyecto de desarrollo de software.

El proceso de traducción del modelo construido en LMR a lenguaje XML puede llegar a ser automatizado mediante un intérprete del modelo LMR, tal como se hizo con el intérprete de GUI, para que de esta forma todo el proceso de codificación sea automático.

Es deseable, que en un trabajo posterior, partiendo de las bases metodológicas entregadas en este documento, se defina la especificación detallada fase a fase de la metodología para desarrollo de software ágil dirigido por modelos a partir de LMR.

VII. Formación de recurso humano

Los resultados de esta investigación serán utilizados para presentar un trabajo de grado de Magister en Ingeniería de Sistemas y Computación en la Universidad Tecnológica de Pereira.

VIII. Referencias

[1] T. Hildenbrand, F. Rothlauf, M. Geisser, A. Heinzl, T. Kude, “Approaches to collaborative software development”, CISIS Conference, IEEE(2008) 523 – 528

[2] “Agile Manifesto” [online] 2001. Disponible en: http://agilemanifesto.org/iso/es/

[3] J.L. Canóvas Izquierdo, J. Cabot, “Creación colaborativa de lenguajes específicos de dominio. Jornadas de Ingeniería de Software y Bases de Datos” [online]. 2012. Disponible en: http://hal.inria.fr/hal-00716432/

[4] C. Casanova, “Factores clave para el éxito o el fracaso en proyectos de implementación de sistemas ERP” [online]. Pectrotecnia, 2010. Disponile en: http://www.petrotecnia.com.ar/abril10/Sin/Factores.pdf

[5] K. Mendes Calo, E. Estevez., P. Fillottrani, P, “Evaluación de metodologías ágiles para desarrollo de software”, Workshop de Investigadores en Ciencias de la Computación WICC2010, Argentina.

[6] M. Daniele, M. Ariel Uva, P. Martelloto, P. “Hacia una automatización de los procesos de desarrollo de software”, Workshop de Investigadores en Ciencias de la Computación WICC2010, Argentina.

[7] P. Vásquez, R. Giandini, P. Bazán, (2010) “Lenguajes notacionales para modelado de procesos: Un análisis comparativo”, Workshop de Investigadores en Ciencias de la Computación WICC2010, Argentina.

[8] S. Martinez, “Estudio comparado de lenguajes gráficos de especificación de sistemas” Proyecto de Titulación Magíster en Ingeniería de Software, Universidad de La Plata, La Plata, Argentina. Dic. 2012

[9] N. Chomsky, “Three models for the description of language”, IRE Transactions on Information Theory, vol. 2(3), pp 113–124. 1956

[10] M. Pulgarín, N. Duque, “Sistema de información para gestionar y administrar colegios oficiales” [online] 2013. Disponible en: http://collage.intelmance.com/

[11] M. Pulgarín, N. Duque, “Sistema de información para gestionar y administrar el envío de publicidad por lotes vía email” [online] 2013. Disponible en: http://199.195.250.24/