Investigación
doi: http://dx.doi.org/10.16925/in.v19i20.1417

Lenguaje de dominio específico para configuración de dispositivos de redes

Daniel Felipe Garzón-Triana[1], Carlos Enrique Montenegro-Marín[2], Paulo Alonso Gaona-García[3]

[1] Estudiante de Ingeniería de Sistemas
[2] Doctor en Ingeniería de Sistemas. Profesor. Correo electrónico: cemontenegrom@udistrital.edu.co
[3] Doctor en Ingeniería de Sistemas. Profesor asociado
Facultad de Ingeniería, Universidad Distrital Francisco José de Caldas, Bogotá, Colombia

Recibido: 26 de enero del 2016 **Aprobado:** 25 de mayo del 2016

Cómo citar este artículo: D. F. Garzón-Triana, C. E. Montenegro-Marín, P. A. Gaona-García, "Lenguaje de dominio específico para configuración de dispositivos de redes", *Ingeniería Solidaria*, vol. 12, no. 20, pp. 83-94, oct. 2016.
doi: http://dx.doi.org/10.16925/in.v19i20.1417

Resumen. *Introducción*: este trabajo hace parte del proyecto "Modelo de algoritmo para implementación de configuraciones en dispositivos de redes", adscrito a la Maestría de Ciencias de la Información y las Comunicaciones de la Universidad Distrital Francisco José de Caldas en el 2015 y 2016. El problema detectado es la búsqueda de configuraciones en dispositivos de red, la cual resulta ser una tarea compleja y repetitiva debido a la gran variedad de configuraciones existentes. *Metodología*: este artículo presenta el desarrollo de dos lenguajes de dominio específico (DSL) basados en MDE, uno gráfico: CNPGraph, y uno textual: CNPText, como propuesta de solución al problema. Cada herramienta genera un documento con las configuraciones de los protocolos de red requeridas para los dispositivos de red, ya sean *routers* o *switches*. *Resultados*: las pruebas se hicieron sobre diferentes escenarios y configuraciones, comparando los tiempos de búsqueda mediante el uso de lenguajes de dominio específico y búsquedas convencionales. Los resultados mostraron que CNPGraph y CNPText reducen en más de un 80 % el tiempo necesario para encontrar las configuraciones. *Conclusiones*: la característica principal, tanto de CNPGraph como de CNPText, es la posibilidad de ampliación del número de protocolos y marcas de fabricantes de *routers* y *switchess*, así como la vinculación de protocolos, con el propósito de añadir nuevos dispositivos de red de diferentes fabricantes y disponer de toda la configuración necesaria para establecer cualquier tipo de red.

Palabras clave: dispositivos de red, ingeniería dirigida por modelos, lenguaje de dominio específico, protocolos de red, Sirius, xText.

p-ISSN 1900-3102 / e-ISSN 2357-6014

doi: http://dx.doi.org/10.16925/in.v19i20.1417

Domain-specific language for the configuration of network devices

Abstract. *Introduction:* This research is part of the project "Algorithm model for the implementation of configurations on network devices", ascribed to the Master's Degree of Information and Communications Sciences of the District University Francisco Jose de Caldas during the years 2015 and 2016. The detected issue is the search of configurations in network devices, which has proven to be a complex and repetitive task due to the wide variety of existent configurations. *Methodology:* This article presents the development of two domain-specific languages (DSL) based on MDE. One is graphic: CNPGraph, and the other is textual: CNPText, as a proposed solution to the problem. Each tool generates a document with network protocols configurations required for network devices, regardless of the devices being *routers* or *switches. Results:* The tests were conducted using different scenarios and configurations, and comparing the search times through the use of domain-specific languages and conventional searches. The results show that CNPGraph and CNPText reduce in over 80 % the time used in finding the required configurations. *Discussion:* The main characteristic of both CNPGraph and CNPText is the possibility of increasing the amount of protocols and private brands of routers and switches, as well as associating protocols with the purpose of adding new network devices from different manufacturers and to access the necessary configuration in order to stablish every type of network.

Keywords: network devices, engineering directed by models, domain-specific languages, network protocols, Sirius, xText.

Linguajem de domínio específico para configuração de dispositivos de redes

Resumo. *Introdução*: este trabalho faz parte do projeto "Modelo de algoritmo para implementação de configurações em dispositivos de redes", do Mestrado de Ciências da Informação e as Comunicações da Universidade Distrital Francisco José de Caldas no ano 2015 e 2016. O problema evidenciado é a busca de configurações em dispositivos de rede, que é uma tarefa complexa e repetitiva por causa da grande variedade de configurações existentes. *Metodologia*: este artigo apresenta o desenvolvimento de duas linguagens de domínio específico (DSL) baseados em MDE, um gráfico: CNPGraph, e um textual: CNPText, como proposta de solução ao problema. Cada ferramenta gera um documento com as configurações dos protocolos de rede requeridos para os dispositivos de rede, sejam eles *routers* ou *switches*. *Resultados*: os testes realizados sobre diferentes cenários e configurações, comparando os tempos de busca através do uso de linguagens de domínio específico e buscas convencionais. Os resultados mostraram que CNPGraph e CNPText reduzem em mais de 80 % o tempo requerido para achar as configurações. *Conclusões*: a principal característica, tanto de CNPGraph quanto de CNPText, é a possibilidade de ampliação do número de protocolos e marcas de fabricantes de *routers* e *switchess*, bem como a vinculação de protocolos, com o intuito de adicionar novos dispositivos de rede de diferentes fabricantes e dispor de toda a configuração requerida para estabelecer qualquer tipo de rede.

Palavras chave: dispositivos de rede, engenharia dirigida por modelos, linguagem de domínio específico, protocolos de rede, Sirius, xText.

1. Introducción

Desde los inicios de la informática, los expertos, desarrolladores e investigadores de *software* han generado abstracciones para programar en términos de su diseño, con la finalidad de brindar facilidades a tareas comunes o de alta complejidad.

La ingeniería dirigida por modelos [1] pretende dar solución a problemas en los cuales existen diversidad de técnicas, lenguajes y mecanismos de codificación. Un ejemplo de ellos es Framework Talisman MDE, un *framework* creado para desarrollar toda clase de aplicaciones usando las bases de la ingeniería dirigida por modelos (MDE). La ventaja de este *framework* es que los usuarios no tienen que tener un conocimiento teórico de MDE, pueden enfocarse en la herramienta que el *framework* les da para desarrollar *software* de mejor calidad en menos tiempo [2]; también esta AIDSL, que es un DSL que define soluciones basadas en el aprendizaje automático que trabaja con un IDE llamado AIIDE. El objetivo principal de la creación de AIDSL fue dar el primer paso hacia la creación de una plataforma basada en estándares para la definición y abstracción de soluciones basadas en el aprendizaje automático de una manera simple y común [3]. Por otro lado, está UML-based AMF Configuration Language [UACL], que es el Marco de Gestión de Disponibilidad (AMF, por sus siglas en inglés) definido por el foro SA para la gestión de aplicaciones de alta disponibilidad. Este necesita configuraciones de aplicaciones que constan de diversas entidades organizadas de acuerdo con el reglamento y las restricciones de la AMF.

Debido a la gran cantidad de entidades involucradas, la creación de estas configuraciones se torna difícil. UACL es una extensión de UML que proporciona a los desarrolladores herramientas para diseñar, editar y analizar las configuraciones de AMF [4]. ANN es un lenguaje de dominio específico para el diseño y validación efectiva de anotaciones de Java. Si bien las anotaciones de Java son una herramienta importante para la adición de metadatos, el soporte nativo que proporciona es muy limitado. ANN nace como una herramienta para superar estas deficiencias y explicitar el modelo conceptual encontrado detrás de las anotaciones [5]; SETT es una herramienta sencilla y útil para los usuarios finales de lenguajes de modelado de dominio específico (DSML). Es un marco de pruebas integrado dentro de un secuenciador DSML.

El secuenciador, que hace parte del sistema de adquisición de datos DEWESoft, genera soporte al desarrollo de las pruebas basadas en modelos que utilizan un alto nivel de abstracción [6]. ModeL4CEP consiste en dos herramientas DSML para facilitar las definiciones de dominio del procesamiento de eventos complejos por expertos de dominio, y para definir patrones de eventos por usuarios no expertos en tecnología. ModeL4CEP nace de la necesidad de los interesados expertos en negocios, de usar Complex Event Proccesing (CEP) sin tener el conocimiento de programación Event Processing Language (EPL). Por medio de ModeL4CEP, los expertos de dominio pueden definir los tipos de eventos relevantes y los patrones dentro de su dominio de negocio, sin la necesidad de ser expertos en programación EPL [7]. Por su parte, DSML4CP aparece en el 2015 como DSLM para la programación concurrente, para trabajar en un nivel de abstracción más alto que el nivel de código.

A partir de un metamodelo con los conceptos y relaciones de los programas concurrentes, la herramienta proporciona un marco para la definición de sintaxis abstracta y sintaxis concreta de DSML4CP. Posterior a las definiciones, la herramienta, a partir de reglas de transformación, genera el código de arquitectura. Para aumentar el nivel de apalancamiento DSML, el mecanismo de transformación de la herramienta soporta Java y C#. Este es el ejemplo más puro del uso de MDE, MDA [8] y de DSL [9], abstracción de un problema en un metamodelo para posterior generación de código [10].

Los DSL se pueden aplicar también en el área de programación de dispositivos de red como *routers*, *switches* o servidores. Típicamente, los DSL buscan reducir tiempo y esfuerzo. Para este caso se va a tomar como escenario de trabajo la programación de diversos dispositivos de red en los cuales cada dispositivo presenta una interfaz de programación propia e incompatible con otras, ya sea por su fabricante o por su protocolo empleado. La propuesta en este artículo es la creación de dos prototipos de lenguajes de programación de dominio específico, uno textual y uno gráfico, que sirvan para la consulta de la configuración de protocolos de red para fabricantes diferentes, cada uno con sus características propias (IP, interfaces, entre otros); de esta manera, se ahorra el tiempo de aprendizaje de estas configuraciones.

2. Metodología

La metodología desarrollada para la resolución de la problemática y comprobación de la hipótesis es un paradigma cualitativo con aporte cuantitativo con enfoque de investigación-acción.

Método de investigación (I-A) [11]: como su nombre sugiere, en ella coexisten en estrecho vínculo el afán cognoscitivo y el propósito de conseguir efectos objetivos y medibles. Este método de investigación considera una investigación como la productora del conocimiento y, por otro lado, se ve la acción.

Las actividades que se desarrollaron para este proyecto fueron las siguientes:

2.1 Investigación general

Se desarrolló un estudio técnico entre las tecnologías existentes de dispositivos de redes. Después se hizo una breve comparación, teniendo en cuenta las configuraciones, proveedores y lenguajes de programación. Se seleccionó y delimitó el proyecto, dependiendo de las equivalencias encontradas en la comparación; además, se establecieron protocolos, tipologías y posibles plataformas de desarrollo.

Para empezar a tomar acción frente al problema por resolver, se procura crear una base de datos con las configuraciones de los dispositivos de red. Estas configuraciones deben ser agrupadas dependiendo de las características que determine el estudio técnico. Luego de haber agrupado estas configuraciones, debe crearse un metamodelo que permita unificar criterios de codificación.

2.2 Acción

Se crearon dos herramientas DSL [12] (Domain Specific Language), una gráfica [13] y una textual [14], que utilicen el metamodelo. Por último, se debe probar lo realizado con el metamodelo en cada dispositivo.

2.3 Análisis de resultados

Basados en los resultados de las configuraciones generadas por las herramientas, se realizó un análisis de los resultados.

2.4 Elaboración de documento

Se realizó el artículo con los resultados, análisis y respectivos prototipos de lenguaje de dominio específico para consulta de configuraciones de dispositivos en redes, como estándar basado en ingeniería dirigida por modelos. Para la elaboración de los dos prototipos, se definieron los mismos protocolos y los mismos fabricantes, para hacer una comparación objetiva de los resultados arrojados por ambas herramientas. La cantidad de protocolos y marcas usadas por las herramientas puede incrementar a medida que en el proceso de desarrollo, se estos se vayan agregando. Los dos DSL cumplen con el principio abierto cerrado (OCP) [15]. Para el caso de las pruebas, se definieron los protocolos NAT [16], STP [17] y 802.1Q [18], dado su grado de complejidad de aprendizaje frente a protocolos de enrutamiento más populares, como RIP [19] u OSPF [20], configurados sobre dispositivos de fabricantes Cisco [21], Mikrotik [22] y Linksys [23], debido a las fuertes diferencias presentadas en los formatos de configuración entre uno y otro.

El objetivo principal de las dos herramientas es reducir el tiempo de búsqueda y aprendizaje de los protocolos de red, así como los recursos necesarios para realizar dichas actividades. Para esto, las herramientas generan, cada una, un documento con las configuraciones solicitadas por el usuario para *routers* y otro documento con las configuraciones requeridas para *switches*. Para tener la certeza de que las configuraciones arrojadas por las herramientas son correctas, se enseñan más adelante pruebas del uso de estas configuraciones sobre simuladores virtuales.

2.4.1 CNPGraph

CNPGraph es el nombre de la herramienta MDE con sintaxis gráfica propuesta en este documento. Está creada con SIRIUS y basada en EMF [24] y GMF [25]. SIRIUS es un generador mediante el cual se definen editores gráficos, puntos de vista de diagramas, tablas y árboles [26]; se define, además, como el lenguaje de creación de CNPGraph, dado que permite un alto nivel de personalización y permite evaluar los cambios en tiempo real.

La solución del problema por medio de CNPGraph inicia con la creación de la herramienta y sus

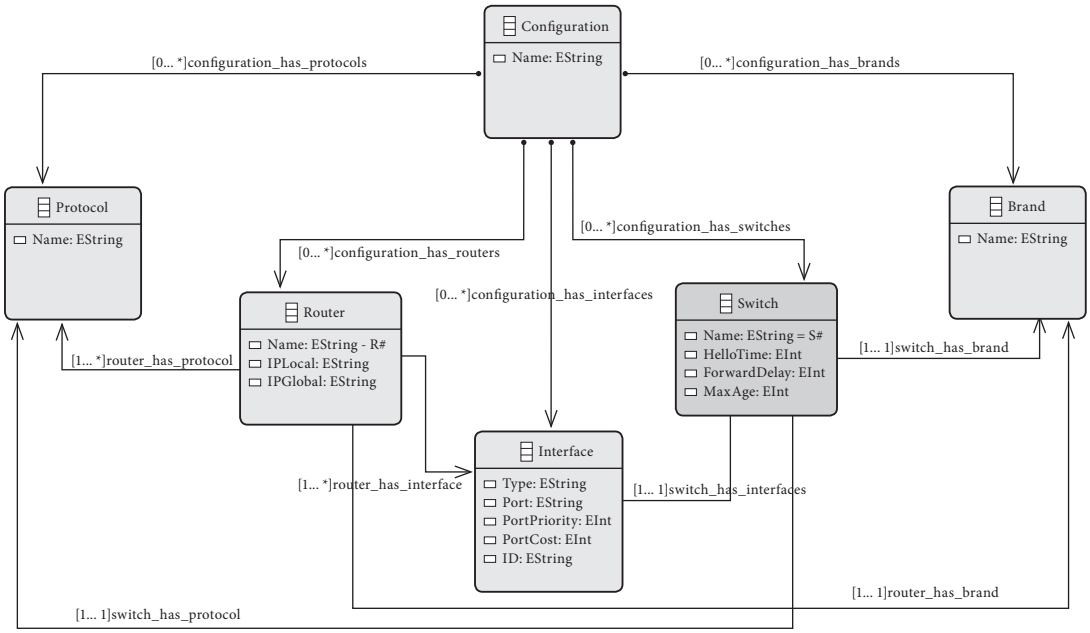


Figura 1. Diagrama de EClases – CNPGraph
Fuente: elaboración propia

posteriores pruebas. Una vez que el funcionamiento de la herramienta sea acertado, se procede a generar el documento con las configuraciones de los protocolos de red. Sobre estas configuraciones se realizan pruebas para corroborar que la información es verídica; de no ser así, se modifica la herramienta con las respectivas correcciones.

Para iniciar con la creación de la herramienta, es necesario partir de un proyecto EMF en eclipse; a partir de este nacen dos elementos, un .ecore (modelo de dominio definido usando Ecore) y un .aird (representación del punto de vista). Estos dos elementos están correlacionados entre sí. En el .aird se define un modelo de EClases, el cual es la base de los elementos existentes en el proyecto.

El modelo de las EClases, mostrado en la figura 1, nace de la abstracción del problema de los tiempos necesarios para el aprendizaje de un indeterminado número de configuraciones de protocolos de red sobre dispositivos de fabricante diferente. Aquí el usuario puede tener cualquier cantidad de dispositivos, en el caso de la herramienta, cualquier cantidad de *routers* y *switches*, de protocolos, de marcas. Además, los *routers* y *switches*, dependiendo del modelo, varían la cantidad de interfaces que puedan tener. Una configuración depende estrictamente del protocolo aplicado sobre el dispositivo de cierta marca y, en ocasiones, de las interfaces o puertos que tenga el dispositivo.

Una vez definidas las entidades, fue necesario concretar los atributos de cada una y las relaciones existentes entre ellas. Para el caso de "Configuration", "Protocol" y "Brand", su único atributo es "Nombre", siendo este una cadena de caracteres. Los *routers*, además de tener un nombre como todos los demás (se sugiere que siga el formato 'R' acompañado de un número), tienen dos IP, una global y una local. Los *switches* siguen el formato de los *routers* con el nombre, con la diferencia de que cambia la 'R' por una 'S'; además, los *switches* tienen como atributos los tiempos "Hello", "Forward" "Delay" y "Max age", todos del tipo entero. Finalmente, el atributo de las interfaces son las características de los puertos que puedan tener los *routers* y *switches*; el tipo de puerto (serial, fast, giga…), el número del puerto (0/0, 1/0/1, por ejemplo), un ID y el costo y prioridad del puerto. En cuanto a las relaciones, se definió que una configuración está compuesta por las demás entidades; los *routers* y los *switches*, tienen, cada uno, un solo protocolo y una sola marca, mientras que sí pueden tener una o más interfaces. Teniendo el diagrama de Eclases se pasa a crear el modelo de dominio definido, es decir, el '.ecore', como se muestra en la figura 2a.

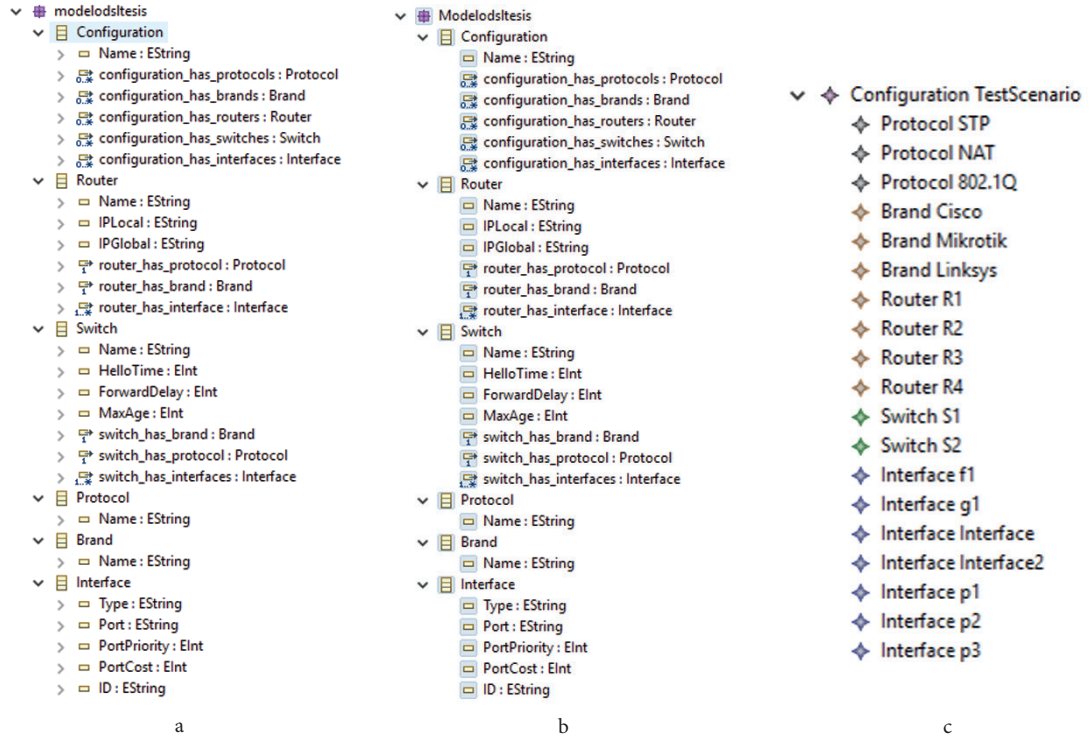


Figura 2. a) Modelo .ecore de CNPGraph; **b)** modelo de generación de código de CNPGraph; **c)** metamodelo CNPGraph
Fuente: elaboración propia

Este .ecore es una representación del diagrama de Eclases a manera de árbol; los dos elementos son iguales (mismas entidades, atributos y relaciones). Es necesario crear un modelo de generación de código .genmodel, que esté basado en el .ecore. El nuevo modelo contiene toda la información del modelo y su generación. La vista del .genmodel, mostrada en la figura 2b, es similar a la del .ecore.

El modelo .ecore es la base del *plug-in*, es decir, de CNPGraph. Se crea entonces un nuevo proyecto y en él un nuevo modelo que tenga las características definidas anteriormente en el proyecto EMF. En este proyecto se crea el metamodelo, como se ve en la figura 2c, en la cual se definen los elementos por defecto con propiedades específicas y relaciones definidas que presenta la herramienta.

SIRIUS es utilizado para crear el Viewpoint del metamodelo. En el nuevo proyecto SIRIUS, es necesario agregar las dependencias del proyecto del metamodelo para definir el Viewpoint sobre este. El Viewpoint define un conjunto de representaciones, diagramas, tablas o árboles; en este caso, el Viewpoint define un diagrama. Antes de crear el diagrama se definen las propiedades del ID y de la extensión que se le va a asignar al modelo. Luego se agrega una representación del tipo diagrama y se establece la clase dominio, que es la clase de la cual hereda o la que está representando. Por medio de *layers*, SIRIUS muestra los elementos relevantes del diagrama, como los nodos y relaciones. Asimismo, sirius permite crear las representaciones de los elementos que hacen parte del metamodelo, ya sean figuras geométricas básicas o imágenes definidas por el desarrollador. Además de las representaciones, por medio de SIRIUS es posible establecer herramientas de creación de nodos y relaciones, con el fin de que el usuario pueda agregar *routers* y *switches* para el caso de CNPGraph. Por último, es necesario agregar el Viewpoint al proyecto para que tome las características definidas con el generador de editores gráficos; una vez definidas todas las características, representaciones, herramientas y relaciones, los DSL estarán terminados. CNPGraph se ve como se presenta en la figura 3.

Esta herramienta permite crear *routers*, *switches* e interfaces, a los cuales se les asigna un protocolo de red y una marca de fabricante establecidos anteriormente en el metamodelo. En el ejemplo

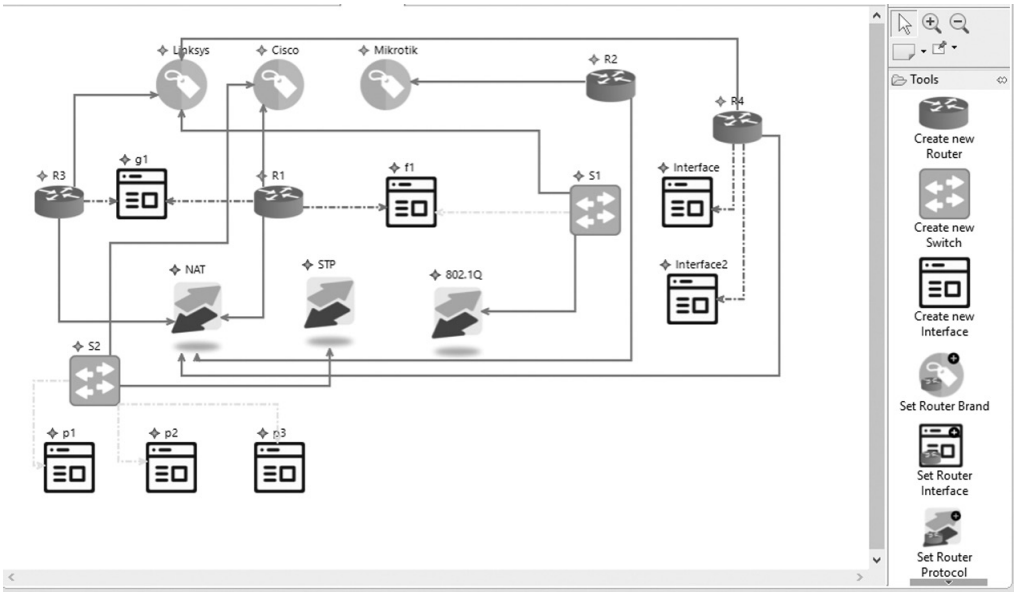


Figura 3. Vista funcional de CNPGraph
Fuente: elaboración propia

mostrado en la figura 3, el *router* R1 tiene como marca a Cisco y como protocolo a NAT. Para la consulta de las configuraciones, en la herramienta se creó un proyecto Acceleo para convertir el modelo en texto mediante plantillas. El proyecto genera dos archivos: Routers.txt y Switches.txt, con las configuraciones de los *routers* y *switches* definidos en el Viewpoint (ver figura 4).

2.4.2 CNPtext

CNPText es el nombre de la herramienta MDE con sintaxis textual propuesta en este documento; está creada con Xtext [27], que genera varios artefactos automáticamente a partir de una gramática en Extended Backus-Naur Form o EBNF. Para la creación de la herramienta se utiliza un parser [28], un metamodelo .ecore y un IDE para eclipse.

El proceso de la solución del problema usando CNPText es el mismo utilizado con CNPGraph; se inicia con la creación de la herramienta, y se realizan pruebas de funcionalidad y de generación de los archivos con las configuraciones de los protocolos de red. Estas configuraciones también son puestas a prueba y en caso de no ser las correctas, se realizan las correcciones necesarias para que lo sean.

El primer paso para la formación de la herramienta es crear la gramática del lenguaje usando Xtext, es decir, el conjunto de palabras reservadas

Routers.txt | Switches.txt

```
Router R1, Brand Cisco:
Protocol NAT

R1(config)#ip nat inside source static 1.1.1.1 192.168.1.5
R1(config)#interface fa 1/0
R1(config-if)#ip nat inside
R1(config-if)#exit
R1(config)#interface se 1/1/1
R1(config-if)#ip nat outside
R1(config-if)#exit
R1(config)#exit

Router R2, Brand Mikrotik:
Protocol NAT

/ip address add address=1.1.1.1/65 interface=Public
/ip firewall nat add chain=dstnat dst-address=1.1.1.1 action=dst-nat
/ip firewall nat add chain=srcnat src-address=192.168.1.2 action=src
/ip firewall nat add chain=dstnat dst-address=1.1.1.1 dst-port=21 pr
/ip firewall filter add chain=forward connection-state=established,r

Router R3, Brand Linksys:
Protocol NAT

vim /proc/sys/net/ipv4/ip_forward
iptables -F
iptables -t nat -F
iptables -t nat -A POSTROUTING -s 1.2.3.4/25 -d 1/1/1 -j MASQUERADE
iptables -A FORWARD -s 1.2.3.4/25 -j ACCEPT
iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
```

Routers.txt | Switches.txt

```
Switch S1, Brand Linksys:
Protocol 802.1Q

Configuration still in progress. Check the new version.

Switch S2, Brand Cisco:
Protocol STP

S2(config)#spanning-tree uplinkfast
S2(config)# spanning-tree hello-time 10
S2(config)# spanning-tree forward-delay 15
S2(config)# spanning-tree max-age 20
S2(config)#int serial 1/1
S2(config-if)# spanning-tree port-priority 2
S2(config-if)# spanning-tree port-cost 1
S2(config)#int fastethernet 2/2
S2(config-if)# spanning-tree port-priority 3
S2(config-if)# spanning-tree port-cost 2
S2(config)#int gigaethernet 3/3/3
```

Figura 4. Archivos autogenerados Routers.txt y Switches.txt
Fuente: elaboración propia

que pueden ser utilizadas para desarrollar utilizando CNPText, como se ve en la figura 5. La herramienta ofrece un conjunto de plantillas y de ayudas para facilitar su aprendizaje y su uso. El formato de la gramática es bastante básico, se inicia haciendo uso de la palabra reservada "Tipo" acompañada de "*Router*" o "*Switch*", dependiendo de cuál dispositivo se va a configurar. Después, mediante la palabra "Código" se define el código del dispositivo (ej. R1, S2), y la palabra "Marca" se usa para definir cualquiera de las marcas determinadas en la lista de la gramática y "Protocolo", seguido de la selección del protocolo que se desee configurar. Ese es el formato base de las configuraciones. Debido a que cada fabricante presenta propiedades y atributos diferentes para cada protocolo, es necesario definir estas características después de haber seleccionado el protocolo. Para esto, CNPText ofrece un asistente para guiar al usuario con la programación y el código necesario para definirlas. La gramática es creada en un archivo .xtext (ver figura 5).

```
grammar org.xtext.routers.tesis.DSL with org.eclipse.xtext.common.Terminals

generate dSL "http://www.xtext.org/routers/tesis/DSL"

Translator:
    configurations+=Configuration*;

Configuration:
    Router | Switch;

Router:
    'Tipo' tipo = "Router"
    'código' name = ID
    'marca' marca = ProtocoloRouter;

Switch:
    'Tipo' tipo = "Switch"
    'código' name = ID
    'marca' marca = ProtocoloSwitch
    ;
```

Figura 5. Creación de la gramática de CNPText
Fuente: elaboración propia

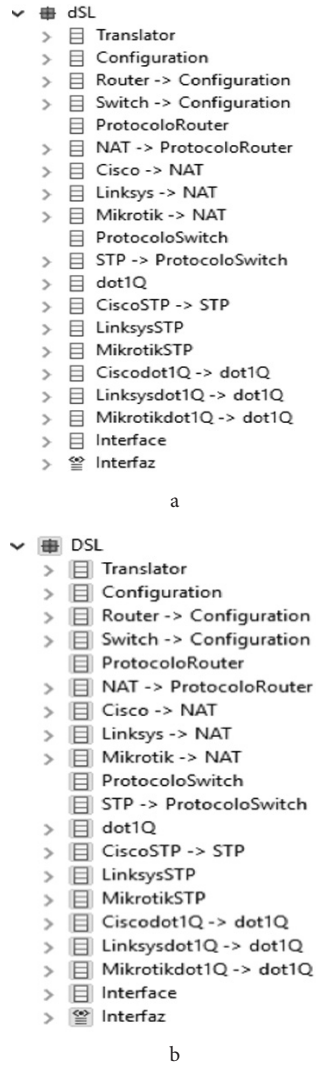


Figura 6. a) modelo .ecore de CNPText y **b)** modelo de generación de código de CNPText
Fuente: elaboración propia

De esta gramática se genera el metamodelo, un archivo .ecore que contiene las entidades, atributos y relaciones utilizadas en la herramienta. Al igual que en la herramienta gráfica, se necesita un modelo generador de código, basado en el metamodelo, un .genmodel. Este proceso se ve en las figuras 6a y 6b.

Después de haber establecido, mediante Xtext, las validaciones de la gramática, reglas de formato, de colores, de interfaz, de personalización, plantillas de ayuda y generación de texto, se generó el *plug-in* de la herramienta para poder ser usada. La representación de la herramienta en uso se muestra en la figura 7.

```
Tipo Router código R9 marca Linksys protocolo NAT
     IP 19.12.1.1/24
     interface  9/9

Tipo Router código R10 marca Cisco protocolo NAT
     IP Local 192.1.1.1/24
     IP Global 1.1.16.1/24
     interface fastethernet 0/0
     interface serial 1/1/0

Tipo Switch código S1 marca Cisco protocolo STP
     hello time 2 forward delay 15 max age 20
     Interface fastethernet 1/0 port priority 32 port cost 19
     Interface serial 0/1/0 port priority 27 port cost 18
```

Figura 7. Vista funcional de CNPText
Fuente: elaboración propia

Como un *plug-in* de eclipse, la herramienta puede ser utilizada para programar el tipo de dispositivo, protocolos, marcas y características de cada uno.

Finalmente, CNPText genera automáticamente dos archivos con las configuraciones de los *routers* y *switches*, programados por el usuario, al igual que en CNPGraph (ver figura 4).

2.5 Pruebas

Los artefactos de *software* generados por ambos prototipos (CNPGraph y CNPText) son documentos de texto con un directorio de configuraciones de los *routers* y *switches* definidos con las herramientas. Como escenario de prueba, en las dos herramientas se definieron *routers* de marcas Cisco, Mikrotik y Linksys, para el protocolo NAT, y *switches* de marca Cisco para el protocolo STP. Las aplicaciones CNPGraph y CNPText, al igual que las pruebas ejecutadas, pueden ser consultadas en http://giira.udistrital.edu.co/DSLs-CNP.zip. Estas aplicaciones dieron los siguientes resultados:

Tabla 1. Resultados escenarios pruebas DSL

Escenario	Routers		Switches	
	CNPGraph	CNPText	CNPGraph	CNPText
Protocolo NAT para CISCO	✔	✔	N/A	N/A
Protocolo NAT para Linksys	✔	✔	N/A	N/A
Protocolo NAT para Mikrotik	✔	✔	N/A	N/A
Protocolo STP para CISCO	N/A	N/A	✔	✔

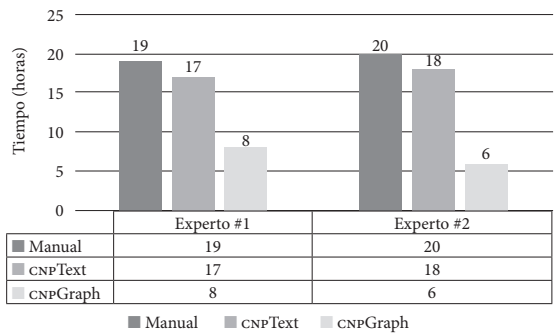
✔: Prueba satisfactoria; X: prueba fallida; N/A: no aplica.
Fuente: elaboración propia

3. Resultados y validación

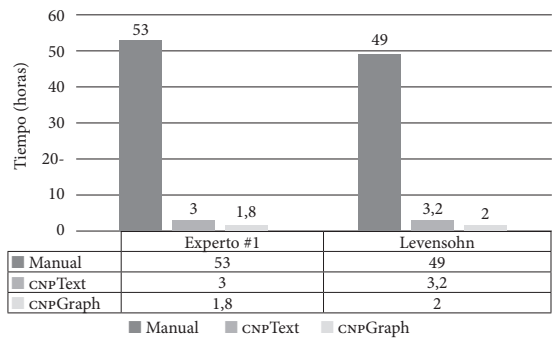
El proceso de validación de las herramientas consiste en la comparación de tiempos empleados en la solución del mismo escenario problema, usando CNPGraph, CNPText y búsqueda manual. Un mismo problema debe ser resuelto mediante los tres métodos mencionados y en cada método se debe realizar la toma de tiempos empleados para una posterior comparación de registros.

Para validar las herramientas, se tomó una muestra de dos personas en un grupo de diez expertos en el área de seguridad en telecomunicaciones y protección de marca, de la empresa Authentic Vision [29]. Los dos expertos realizaron dos grupos de pruebas para la validación; el primero consistía en aprender dos protocolos de red sobre tres fabricantes diferentes de forma manual, investigando por su propia cuenta y después usando las herramientas CNPGraph y CNPText. El segundo grupo consistía en aprender quince protocolos de red sobre cinco dispositivos de diferente fabricante —al igual que para el grupo 1—, una prueba a manera manual y la otra usando los DSL creados para el proyecto.

Para cada prueba se tomaron los tiempos de aprendizaje y testimonios, en cuanto a la facilidad de acceso a la información. Una vez finalizadas las actividades, los resultados se representaron en las figuras 8a y 8b.



a



b

Figura 8. a) Resultados grupo de pruebas 1; **b)** resultados grupo de pruebas 2
Fuente: elaboración propia

Los resultados del grupo de pruebas 1 demuestran que para un número de configuraciones pequeñas el uso de la herramienta textual, sumado al tiempo de aprendizaje, no difiere sustancialmente en el tiempo utilizado en el proceso de búsqueda manual, pero sí reduce el tiempo de búsqueda aproximadamente en un 10 %. Por el otro lado, el uso de la herramienta gráfica, sumado al tiempo de aprendizaje, reduce entre el 60 y 70 % el tiempo necesario para conocer las configuraciones de los protocolos de red sobre *routers* y *switches*, frente al tiempo utilizado en el proceso de búsqueda manual, siendo CNPGraph la mejor alternativa para solucionar el problema.

Los resultados del grupo de pruebas 2, en el cual el escenario problema presenta una gran cantidad de configuraciones, varían en los tiempos de comparación, debido a que ya no es necesario sumar el tiempo de aprendizaje, y sirve como escenario en el cual el usuario conoce cómo funcionan las herramientas. Haciendo uso de CNPText, el tiempo empleado para solucionar el problema es apenas de un 6 % aproximadamente, del tiempo gastado realizando las búsquedas de manera manual. Mientras que usando CNPGraph el tiempo se reduce en un 96 % frente al tiempo de búsqueda manual, lo que permite que sea la herramienta más cómoda y rápida de las tres. Los tiempos de solución brindados por CNPText y CNPGraph son bastante similares, una vez se conoce cómo utilizar las herramientas; en caso contrario, CNPGraph resulta ser mejor opción gracias a su naturaleza deductiva, lo que reduce los tiempos de aprendizaje.

4. Discusión

En este apartado se presenta una discusión respecto al desarrollo, las pruebas y la validación de las herramientas construidas, además de su facilidad de uso.

La característica principal, tanto de CNPGraph como de CNPText, es la posibilidad de ampliación del número de protocolos y marcas de fabricantes de los *routers* y *switches*. Está proyectado —además de continuar agregando más protocolos y marcas— añadir nuevos dispositivos de red diferentes a *routers* y *switches*, con el fin de conocer toda la configuración necesaria para establecer cualquier tipo de red. Por otro lado, en un futuro se evaluará la posibilidad de cambiar el formato de los documentos generados con las configuraciones, con el fin de crear archivos con el formato que el dispositivo reciba como entrada, para transformar CNPText y CNPGraph en herramientas no solo de consulta, sino también de configuración directa.

CNPGraph es una herramienta sencilla, de poco tiempo de aprendizaje, ya que su naturaleza gráfica la hace deductiva. Por su parte, CNPText es una herramienta de fácil aprendizaje también, aunque de mayor complejidad que su hermana CNPGraph, debido a que es necesario aprender el código y sintaxis de este lenguaje de programación.

Para un número menor a diez configuraciones, el tiempo de aprendizaje de los protocolos de red utilizando las herramientas CNPGraph y CNPText es muy similar al tiempo de aprendizaje de manera manual mediante investigaciones propias. A medida que el número de configuraciones sea mayor, el tiempo de aprendizaje para la consulta de los protocolos de red utilizando las herramientas CNPGraph y CNPText es significativamente menor que el tiempo necesario para aprender este tipo de configuraciones de manera manual. CNPText y CNPGraph no son herramientas para configurar dispositivos de red, sino para la consulta de las configuraciones de protocolos de red para ser aplicados sobre los dispositivos mediante el uso de otras herramientas. Este documento ha numerado una serie de tareas que deben ser llevadas a cabo en un futuro.

5. Conclusión

El hecho de haber utilizado ingeniera dirigida por modelos permitió tener un metamodelo unificado para configuración de dispositivos de red, lo que evidencia que se puede llegar a una estandarización respecto a los mecanismos de codificación de dispositivos y evitar así la actual diversidad que existe.

La ingeniería dirigida por modelos puede ser aplicada en diversos ámbitos, para ayudar a disminuir la complejidad de uso y aprendizaje por parte del usuario final; también permite simplificar procesos repetitivos de código, y con esto se reducen tiempos y recursos necesarios para llevar a cabo dichos procesos. Además, gracias al uso de modelos independientes de plataformas, el desarrollador

deja a un lado la preocupación por los detalles y por las diferencias entre un lenguaje y otro, ya que por medio de MDA, es posible transformar un proyecto de un lenguaje a otro.

Los DSL son una opción adecuada para generar sintaxis textuales y gráficas, que se centran en el negocio, y dejan de lado la complejidad de aprender un lenguaje de uso genérico y metodologías de desarrollo de *software*, hecho que permite que cualquier persona con conocimientos en el área de los DSL pueda usar la herramienta.

Referencias

[1] J. Fischer, M. Scheidgen, I. Schieferdecker, R. Reed, *SDL: Model-Driven Engineering for Smart Cities*. Berna, Suiza: Springer International Publishing, 2015, pp. 19-26.

[2] V. García-Díaz, J. Barranquero, B. Pelayo, E. Palacios, O. Sanjuan, "TALISMAN MDE Framework: An Architecture for Intelligent Model-Driven Engineering", en *Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living*, S. Omatu, M. P. Rocha, J. Bravo, E. Corchado et al. Eds. Berlín, Alemania: Springer Berlin Heidelberg, 2009, pp. 299-306.

[3] V. García-Díaz, J. Pascual-Espada, C. Pelayo Bustelo y J. M. Cueva-Lovelle, "Towards a Standard-based Domain-specific Platform to Solve Machine Learning-based Problems", *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 3, n.º 5, pp. 6-12, dic. 2015 [en línea]. Disponible en: http://www.ijimai.org/journal/sites/default/files/files/2015/11/ijimai20153_5_1_pdf_15294.pdf

[4] P. Salehi P., A. Hamou-Lhadj, M. Toeroe y F. Khendek, "A UML-based domain specific modeling language for service availability management: Design and experience", *Computer Standards & Interfaces*, vol. 44, pp. 63-83, nov. 2015 [en línea]. Disponible en: http://www.sciencedirect.com/science/article/pii/S0920548915001063

[5] I. Córdoba, J. de Lara, "ANN: A domain-specific language for the effective design and validation of Java annotations", *Computer Languages, Systems & Structures*, vol. 45, pp. 164-190, abril 2016 [en línea]. Disponible en: http://www.sciencedirect.com/science/article/pii/S1477842416300318

[6] T. Koz, M. Mernik y T. Kosar, "Test automation of a measurement system using a domain-specific modelling language". *Journal of Systems and Software*, vol. 111, pp. 74-88, nov. 2015. Disponible en: http://www.sciencedirect.com/science/article/pii/S0164121215002058

[7] J. Boubeta-Puig, G. Ortiz y I. Medina-Bulo, "MODEL4CEP: Graphical domain-specific modeling languages for CEP domains and event patterns", *Expert Systems with Applications*, vol. 42, pp. 8095-8110, nov. 2015 [en línea]. Disponible en: http://www.sciencedirect.com/science/article/pii/S0957417415004479

[8] O. Pastor y J. Molina, "The Need for New Development Environments" en *Model-Driven Architecture in Practice*. Berlín, Alemania: Springer, 2007, pp-13-18.

[9] W. Mohamed, *Domain-Specific Languages*. Berlín, Alemania: Springer, 2009, pp. 20-44.

[10] E. Marand, E. Azadi y M. Challenger, "DSML4CP: A Domain-specific Modeling Language for Current Programming", *Computer Languages, Systems & Structures*, vol. 44, pp. 319-341, sept. 2015 [en línea]. Disponible en: http://dl.acm.org/citation.cfm?id=2852437

[11] R. Hernández, C. Fernández y M. Bauptista, *Metodología de la Investigación*, 5ta. Ed., México, D. F., México: McGrawl-Hill, 2010, p. 610 [en línea]. Disponible en: https://www.esup.edu.pe/descargas/dep_investigacion/Metodologia%20de%20la%20investigaci%C3%B3n%205ta%20Edici%C3%B3n.pdf

[12] I. Reinhartz, A. Sturm, T. Clark, S. Cohen y J. Bettin, "Domain-Specific Languages and Standardization: Friends or Foes?", en *Domain Engineering*. Part II, Berlín, Alemania: Springer. 2013, pp. 159-186.

[13] M. Bernardo, V. Cortellessa y A. Pierantonio, "Graph Transformations for MDE, Adaptation, and Models at Runtime", en *Formal Methods for Model-Driven Engineering*. Berlin, Alemania: Springer, 2012, pp.137-191.

[14] A. Vallecillo, J-P. Tolvanen, E. Kindler, H. Störrle y D. Kolovos, "Domain-Specific Textual Meta-Modelling Languages for Model Driven Engineering", en *Modelling Foundations and Applications*. Berlin, Alemania: Springer, 2012, pp. 259-274.

[15] B. Pernici, "On the Semantics of the Extend Relationship in *Use Case Models:* Open-Closed Principle or Clairvoyance?", en *Advanced Information Systems Engineering*. Berlín, Alemania: Springer. 2010, pp. 409-423.

[16] D. Medhi, J. Nogueira, T. Pfeifer y S. F. Wu, "Securing a Path-Coupled NAT/Firewall Signaling Protocol" en *IP Operations and Management*, Berlín, Alemania: Springer, 2007, pp. 61-72.

[17] Z. Cai, C. Wang, S. Cheng, H. Wang y H. Gao, "NC-STP: A High Performance Network Coding Based Space Transport Protocol", en *Wireless Algorithms, Systems, and Applications*. Berna, Suiza: Springer International Publishing, 2014, pp. 34-43.

[18] P. Jungck, R. Duncan y D. Mulcahy, "Packet Information Block and System Packet Operations" en *PacketC Programming*. New York, Estados Unidos: Apress, 2011, pp. 175-204.

[19] C. Carthern, W. Wilson, R. Bedwell y N. Rivera, "Routing", en *Cisco Networks*. Las Vegas, Estados Unidos: Apress, 2015, pp. 93-147.

[20] P. Tadimety, *OSPF: A Network Routing Protocol*. Londres, Inglaterra: Apress, 2015, pp. 12-15.

[21] Cisco, "Cisco Corporate Overview and Resources | The Network", 2016 [en línea]. Disponible en: https://newsroom.cisco.com/overview

[22] Mikrotik, "MikroTik Routers and Wireless: About MikroTik", 2016 [en línea]. Disponible en: http://www.mikrotik.com/aboutus

[23] Belkin, "About Belkin", 2016 [en línea] Disponible en: http://www.belkin.com/us/aboutus/

[24] A. Schürr, M. Nagl y A. Zündorf, "The EMF Model Transformation Framework", en *Applications of Graph Transformations with Industrial Relevance*. Berlín, Alemania: Springer, 2008, pp. 566-567.

[25] B. Malloy, S. Staab y M. van den Brand, "Automated Co-evolution of GMF Editor Models", en *Software Language Engineering*. Berlín, Alemania: Springer. 2011, pp. 143-162.

[26] Eclipse, "Sirius - The easiest way to get your own Modeling Tool", 2016 [en línea]. Disponible en: http://www.eclipse.org/sirius/

[27] S. Efftinge y M. Spoenemann, "Xtext-Language Engineering Made Easy!", 2016 [en línea]. Disponible en: http://www.eclipse.org/Xtext/

[28] R. Heckmann, "An efficient ELL(1)-parser generator", *Acta Informatica*, vol. 23, pp. 127-148, oct. 2011. Disponible en: http://link.springer.com/article/10.1007%2FBF00289494

[29] Authenticvision, "Authentic Vision: About Authentic Vision", 2016 [en línea]. Disponible en: http://www.authenticvision.com/company/about-authentic-vision.html