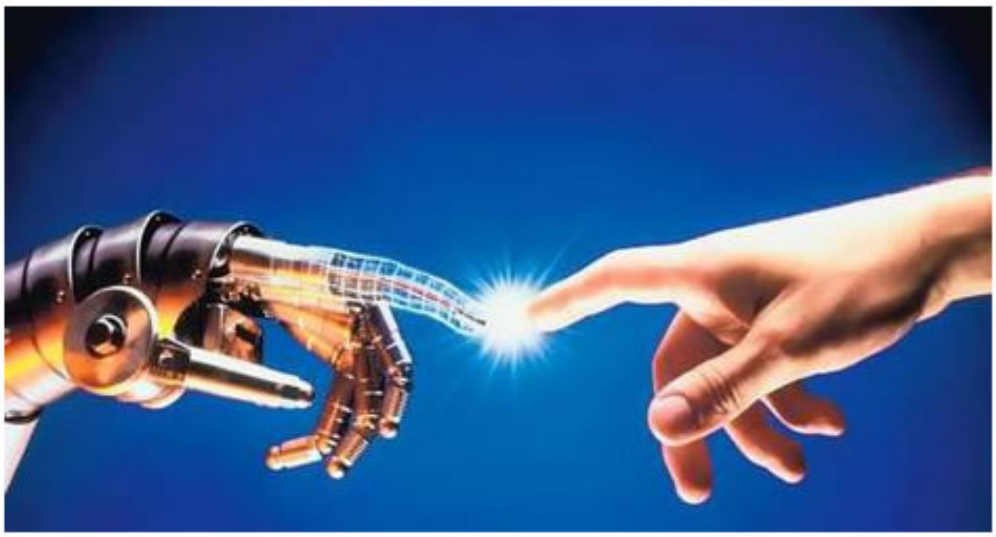
Rama de Estudiantes UNED
http://www.ieec.uned.es/ieee-uned/

BOLETÍN ELECTRÓNICO DE LA RAMA DE ESTUDIANTES DEL IEEE DE LA UNED

IEEE ELECTRONIC JOURNAL OF THE UNED STUDENT BRANCH

ISSN: 1989-2195



Número XII
Edición Julio 2009

Boletín Electrónico Rama de Estudiantes de la UNED
Julio-2009

EDITOR
Sergio Martín
(sergio.martin@ieee.org)

REVISORES
Manuel Castro
Sergio Martín
Germán Carro

DISEÑO PORTADA
Sergio Martín

AUTORES
José Daniel Martínez
Germán Carro
Manuel Blázquez
Pablo Blázquez
Miguel Latorre
Emerson Castañeda
Gloria Murillo.

AGRADECIMIENTOS
Vicerrectorado de Investigación UNED
Vicerrectorado de Estudiantes y Desarrollo Profesional UNED
Escuela Técnica Superior de Ingenieros Industriales UNED
Escuela Técnica Superior de Ingenieros Informáticos UNED
Sección Española del IEEE
Departamento de Ingeniería Eléctrica, Electrónica y de Control (DIEEC) UNED
IEEE Women In Engineering (WIE)

AGRADECIMIENTO ESPECIAL
Agradecemos a nuestro Catedrático de Tecnología Electrónica y Profesor Consejero de la Rama, Manuel Castro, todo el tiempo y la dedicación que nos presta, así como, el habernos dado la posibilidad de colaborar con el Capítulo Español de la Sociedad de Educación del IEEE para la elaboración del mismo.

Agradecemos a todos los autores, y a aquellos que han colaborado para hacer posible este Boletín Electrónico.

BOLETÍN DESARROLLADO EN COLABORACIÓN CON EL CAPÍTULO ESPAÑOL DE LA SOCIEDAD DE EDUCACIÓN DEL IEEE

IEEE
Sociedad Educación
CAPÍTULO ESPAÑOL

Nueva Junta Directiva 2009

Elio San Cristóbal. Nuevo Presidente de la rama de estudiantes IEEE-UNED. Ingeniero Informático y estudiante de doctorado en el DIEEC. Actualmente trabaja en el Departamento de Ingeniería Eléctrica, Electrónica, y Control en proyectos de investigación. elio@ieec.uned.es

Germán Carro. Vicepresidente de la Rama de Estudiantes del IEEE-UNED. Estudiante de Ingeniería Técnica en Informática de Sistemas por la UNED. En años anteriores ha colaborado con la Junta Directiva como Coordinador de Actividades Generales. germancf@ieee.org

Rosario Gil. Secretaria y Tesorera de la Rama de Estudiantes del IEEE-UNED. Coordinadora de Woman In Engineering. Ingeniera de Telecomunicaciones, actualmente trabaja como Becaria de Investigación en el DIEEC. rgil@ieec.uned.es

Gloria Murillo. Coordinadora del Comité del Boletín Electrónico. Ingeniero Técnico en Telecomunicaciones, y estudiante de Ingeniería Industrial por la UNED. lorycordero151@hotmail.com

Ramón Carrasco. Coordinador de Actividades Generales. Licenciado en Ciencias Física, especialidad Electrónica. Director de Colegio Karbo de la Coruña centro de Educación Infantil, Primaria y Formación Profesional de Grado Medio y Superior. moncho@warningcorp.com

Guillermo Lafuente. Coordinador de Actividades Generales. Ingeniero Técnico en informática de sistemas por la UNED. También estoy haciendo la certificación CCNA de Cisco Systems. guiye1984@hotmail.com

Igor Chávez. Coordinador del Grupo de Robótica. Técnico en Electrónica en la National Schools. Actualmente alumno de Ingeniería T. Industrial especialidad Electrónica Industrial de la UNED. igorchavez@ieee.org

Carlos Conde. Estudiante de Ingeniería Técnica en Informática de Sistemas por la UNED. Miembro de la Rama Estudiantil del IEEE-UNED y Coordinador del Grupo de Robótica del mismo. carlosch@mundo-r.com

Rubén Alonso Paredero. Coordinador del Grupo de Robótica. Estudiando 5º de Física. Empecé en la UPV-EHU y continué en la UCM. Trabajo en Barcelona como Técnico Especialista en Robótica, desarrollando el proyecto SEAT-EXEO. rubenalonso78@hotmail.com

José Antonio Cámara. Coordinador Grupo de Control de Procesos. Ingeniero T. industrial, especialidad en electrónica industrial en la universidad de Alcalá, estudiante de ingeniería electrónica (UAH) e ingeniería industrial (UNED), he trabajado en diversos sectores, telecomunicación, automóvil, aerogeneradores y materiales. jcm92251@alu.uah.es

Alberto Dopico. Coordinador Grupo de Software Libre. Estudiante de Ingeniería Técnica Industrial Electrónica en la UNED. alberto.dopico@ieee.org

Pablo Calviño. Coordinador Grupo Diseño Web. Experto en desarrollo web y arquitectura de la información. Técnico Superior en Informática de Gestión y estudiante de Ingeniería de Sistemas por la UNED. Miembro del IAI (Information Architecture Institute). kemosade@gmail.com

Alicia Sánchez Ferro. Coordinadora de Socios y Bienvenida. Ingeniera Técnica Informática. Actualmente trabajo de operadora de sistema
alicia.sanchez.ferro@gmail.com

Sergio Martín Gutiérrez. Anterior presidente de la Rama. Actual Coordinador de Publicaciones y del Boletín Electrónico. Ingeniero Informático. Estudiante de doctorado en el DIEEC en temas de Computación Ubicua, Entornos Inteligentes y Aprendizaje con Dispositivos Móviles.
sergio.martin@ieee.org

Manuel Castro. Profesor Consejero de la Rama de Estudiantes del IEEE-UNED. Catedrático de Tecnología Electrónica. Fellow del IEEE y primer presidente del Capítulo Español de la Sociedad de Educación del IEEE.
mcastro@ieec.uned.es

Eugenio López. Mentor de la rama de estudiantes IEEE-UNED, y antiguo presidente de la rama de Estudiantes del IEEE-UNED. Ingeniero Industrial por ETSII de la UNED, y estudiante de Doctorado en el DIEEC de la Escuela. Actualmente trabaja en Niedax Kleinhuis.
elopez@ieec.uned.es

Índice

ISSN: 1989-2195

Últimas Noticias de la Rama de Estudiantes IEEE-UNED

Germán Carro
Vicepresidente Rama de Estudiantes IEEE-UNED
Universidad Nacional de Educación a Distancia
A Coruña, España
germancf@ieee.org

I. PROYECTO SSETI

El pasado 23 de Junio se dio por terminada la presentación de informes de los equipos que componen este proyecto. Con ello ha finalizado la Fase A; en nuestro equipo al menos; satisfactoriamente. A partir de ahora los encargados de coordinación del proyecto confeccionarán un dossier que nos servirá, a todos los que colaboramos en esta aventura, como punto de partida de la Fase B. En ella, nuestro equipo, pasará de las propuestas en papel al diseño de circuitería e implementación del sistema de comunicaciones que hemos propuesto.

En principio y por los comentarios realizados hasta ahora, se está avanzando según lo previsto. El lanzamiento del satélite se ha planificado para el tercer o el cuarto trimestre del 2010, dependiendo de la ventana de lanzamiento más adecuada en esas fechas.

En breve, tras unas semanas de vacaciones comenzará la siguiente fase de la que puntualmente os iremos informando. No obstante tenéis la información sobre este proyecto y los diferentes equipos que lo componen en:

http://www.sseti.net/index.php?option=com_content&view=category&layout=blog&id=11&Itemid=31

Aprovechamos para destacar y agradecer, especialmente, el trabajo realizado por Alberto Dopico en los desarrollos eléctricos y de frecuencias llevados a cabo durante toda esta fase, a Pablo Herrero, por sus diligentes y oportunos consejos, y al resto del equipo que forma el COMM Team del proyecto SSETI Swarm.

II. GRUPO DE ROBÓTICA

Desde finales de Junio el grupo de robótica ha estado preparando dos equipos para presentarse a las pruebas de la próxima Campus Party de Valencia, concretamente al certamen de este año de la CampusBot. Este evento agrupa a equipos nacionales e internacionales de aficionados y profesionales de la robótica y propone diferentes pruebas para las competiciones que acoge durante varios días.

Las pruebas escogidas por nuestros equipos fueron la del Laberinto y la de los seguidores o Xtreme-robotrackers. Para ello salieron hacia Valencia un grupo desde A Coruña (Galicia) y otro desde Terrasa (Cataluña) con el reto de mostrar las habilidades de nuestra Rama en esta competición, y sobre todo con el objetivo de pasarlo lo mejor posible y aprender lo máximo en esta experiencia sobre robótica.

Los días de celebración de este evento fueron desde el 27 de Julio al 2 de Agosto y el lugar; como no podía ser de otra manera; la Ciudad de las Artes y de las Ciencias de Valencia. Sin duda podemos decir que nuestra presencia allí resultó fructífera. Si bien al final tuvimos que fusionar los dos grupos en uno solo ya que al hardware del Bot de Terrasa se le acabó instalando el software del Bot de Coruña, para así ser más competitivos en las pruebas; un ejemplo perfecto de trabajo en equipo. Esa decisión nos ha dado el 3º puesto en la Prueba del Laberinto. Un meritorio puesto con medalla y diploma incluido.

Respecto a la prueba de los Xtreme-robotrackers la competencia fue bastante fuerte y no pudimos llegar a clasificar por falta de velocidad. No obstante estamos contentos de este resultado en nuestra primera participación oficial como Rama de Estudiantes del IEEE-UNED en un evento de este tipo. ¡Muchas gracias a todos, y en especial a los que habéis estado allí, Ramón Carrasco, Nuria Girbau y Manel Garcia!.

Así mismo, debemos destacar la presencia de Igor Chávez en la participación de robots bípedos. Sin duda una de las pruebas más espectaculares dentro de las de robótica general. La feroz competencia en estas lides tampoco nos ha permitido clasificar, pero sí disfrutar de un rato agradable y divertido siguiendo las evoluciones de su robot en el campeonato.

Tenéis toda la información sobre las pruebas en este enlace:

http://www.campus-party.es/index.php/CampusBot.html

III. EXCURSIONES

Para todos aquellos que estéis interesados, se está coordinando una excursión para visitar las delegaciones de la NASA (www.nasa.es) y el INTA (http://www.inta.es/) en Madrid. Para ello estamos colaborando diferentes Ramas de Estudiantes del IEEE: SB IEEE de la Universidad de Sevilla, SB IEEE de la UNED, SB IEEE de la UPM, y está abierta aún la participación de otras Ramas.

ISSN: 1989-2195

Las fechas para la excursión están previstas para el mes de Octubre, y se intentaría hacer coincidir con un fin de semana para aprovechar el viaje y disfrutar del contacto con las distintas Ramas que se animen a asistir. Es importante realizar una planificación previa para fletar los autobuses desde los distintos lugares donde haya miembros interesados en acudir a este encuentro, con lo que podéis poneros en contacto con Germán Carro o con Ramón Carrasco vía correo electrónico de cara a apuntaros.

Como es la Rama de Sevilla la que está organizando todo el viaje, aún no tenemos datos concretos de precios, lugares y demás, pero conforme nos vayan pasando la información os la iremos haciendo llegar. Por ahora el grupo de Terrasa de nuestra Rama ya ha enviado un listado de miembros interesados en acudir, el resto de grupos y miembros repartidos por los diferentes Centros Asociados de la UNED que formáis parte de nuestra Rama sólo tenéis que decirlo y os iremos incluyendo en esa lista.

IV. WEBINARIOS IEEE

Como muchos de vosotros ya sabréis, el IEEE propone periódicamente varios seminarios on-line, completamente gratis, sobre temas de actualidad. Si bien la elección depende de los intereses académicos o profesionales de cada uno, de los más recientes; que aún se pueden descargar en diferido; a mí me han interesado los relativos a la robótica.

Concretamente los dos primeros webinarios de un grupo de cuatro que versan sobre este tema:

- *I Robotics 101*: http://spectrum.ieee.org/webinar/66231
- *II Sense, Think, Act for Umanned Robotics Systems*: http://spectrum.ieee.org/webinar/65454

Y los próximos webinarios sobre este tema serán:

- *III Robotics in Academia* (17 de Septiembre de 2009)
- *IV Future of Robotics* (12 de Noviembre de 2009)

Así mismo os recordamos que el IEEE Spectrum tiene varias secciones especializadas en diferentes canales científicos (*aeroespace*, *biomedical*, *computing*…) y entre ellos hay uno dedicado en exclusiva a la robótica, y noticias relacionadas con ésta, que podéis ver en este enlace:

http://www.spectrum.ieee.org/robotics

Entretenimiento y formación, un buen incentivo para las tardes de lluvia del verano.

V. CURSOS GRATUITOS DEL MIT

De la mano de la AESS del IEEE (Aerospace and Electronic System Society); una de las muchas "society" del IEEE; se nos informa de la puesta en marcha de un nuevo portal de formación. En él, entre las muchas opciones formativas que aparecen, podemos destacar las referencias a los cursos gratuitos del MIT (Massachusets Institute of Technology), en este enlace os mostramos algunos de ellos:

http://ocw.mit.edu/OcwWeb/web/courses/courses/index.htm

Como podéis ver hay una gran variedad y diversidad para elegir.

VI. CANDIDATURA SBC 2010

Ya se ha iniciado la entrega de información a varios de los patrocinadores institucionales y privados de cara a preparar la candidatura de nuestra Rama para albergar el próximo SBC (Student Branch Congress) del 2010, el evento bianual más importante dentro de la Región 8.

El lugar de celebración sería A Coruña y tanto nuestro Coordinador de Actividades Generales; Ramón Carrasco; como nuestro Vicepresidente; Germán Carro; así como el resto de miembros del grupo que tenemos en esa localidad, han estado visitando y consiguiendo los apoyos necesarios para ello. El propio Ayuntamiento de esta ciudad, la Fundación Caixa Galicia, la Fundación Barrié y por supuesto el propio Centro Asociado de la UNED en esa localidad, ya han dado su conformidad al respecto. Finalmente el esfuerzo se ha centrado en conseguir también el apoyo de empresas privadas y preparar el dossier definitivo que ya se ha remitido a nuestros representantes en el IEEE a nivel Región 8 conteniendo nuestra candidatura.

Para aquellos que no sepáis lo que es el SBC, podéis consultar el enlace del último que se ha celebrado en Londres en el 2008: http://sbc2008.org/

Es un evento que se celebra cada dos años; y comprobareis en qué consiste y cual es el número de Ramas que aglutina teniendo en cuenta que los asistentes proceden de África, Europa, y parte de Asia, fundamentalmente.

VII. COORDINADORA DE SOCIOS:

Queremos agradecer a Alicia Sánchez el que haya empezado a colaborar con nosotros como nuestra nueva Coordinadora de Socios y Bienvenida. Su labor será la de saludar e informar a todos aquellos estudiantes interesados en conocer y formar parte de nuestra Rama del IEEE en la UNED, activar y ayudarles en su acceso a los grupos de aLF, en el registro como miembro del IEEE, y explicarles como pueden participar de nuestras actividades y colaborar con nosotros aportando ideas, sugerencias y comentarios para mejorar nuestra Rama. Así mismo contaremos con su inestimable ayuda para reforzar nuestro grupo y actividades en Madrid. ¡Gracias Alicia y bienvenida!

Software Libre para Simulación Electrónica

José Daniel Martínez Calero
Dep. Ingeniería Eléctrica, Electrónica y de Control
Universidad Nacional de Educación a Distancia
Albacete, España
josedaniel_martinez@yahoo.es
http://electronicalibre.wordpress.com

***Abstract*— En este documento se analizan las herramientas de código abierto y/o software libre para la simulación de circuitos electrónicos y pretende mostrar los principales proyectos de este tipo que existen en la actualidad.**

Keywords- Simulación electrónica; Software libre; Software de código abierto

I. INTRODUCCIÓN

En los últimos años se está observando un mayor uso de software libre y/o de código abierto en todos los ámbitos de la informática. Frente a los programas comerciales de pago presentan grandes ventajas como la libertad de descargártelos y poder usarlos sin ningún tipo de molestia con las licencias. El constante aumento de usuarios de los sistemas GNU/Linux contribuye a una mayor difusión y a un mejor soporte de este tipo de programas.

El desarrollo de herramientas EDA de código abierto es un gran hito para la historia de la informática ya que para desarrollarlas no sólo se requieren conocimientos de programación sino que también se requieren conocimientos en electrónica, en modelos de componentes, en las ecuaciones que rigen el comportamiento de los modelos y en infinidad de algoritmos de cálculo. Hay que tener en cuenta que empresas como *Cadence* o *Synopsys* gastan cantidades ingentes de dinero en investigación y esas cantidades no están disponibles para el desarrollo de programas de código abierto.

En los siguientes párrafos nos centraremos en las dos aplicaciones más importantes de código abierto utilizadas para la simulación de circuitos que existen en la actualidad: el proyecto gEDA y el programa Qucs. En el punto IV hablaremos sobre Fedora Electronic Lab, todo un DVD lleno de herramientas de código abierto.

II. PROYECTO GEDA

El proyecto gEDA (www.gpleda.org) es un conjunto de herramientas EDA (hay herramientas propias del proyecto y herramientas asociadas creadas y/o mantenidas externamente al proyecto), todas de código abierto y para plataformas Linux. El proyecto gEDA nos brinda la posibilidad de crear archivos de netlist para después poder simularlos con alguna de las herramientas de simulación asociadas al proyecto o con otros programas SPICE.

El proceso para poder realizar una simulación con las herramientas del proyecto gEDA consta de 3 pasos: 1) realización del esquema del circuito, 2) obtención del archivo de netlist y 3) simulación y representación de las gráficas de las variables del circuito.

Una de las grandes desventajas del proyecto gEDA es principalmente esa: el no tener agrupadas todas las herramientas en un único programa y la necesidad de tener que usar una herramienta o programa diferente para cada paso.

Para la realización del esquema usaremos el programa *gschem* que se caracteriza por tener gran multitud de símbolos de componentes y al que se pueden añadir más. En la página web http://www.gedasymbols.org/ encontraremos varias bibliotecas con símbolos preparadas para ser usadas con el programa. En la figura 1 se muestra la interfaz del programa.

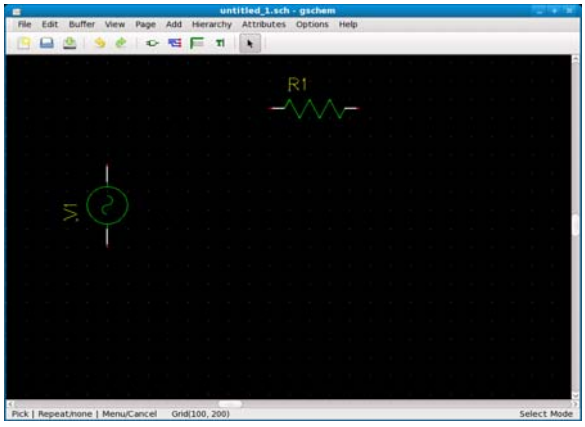


Figura 1. Interfaz del programa gschem

Los símbolos de los componentes vienen agrupados por diversas categorías y se pueden seleccionar de una manera rápida desde la ventana de selección de símbolos (figura 2).

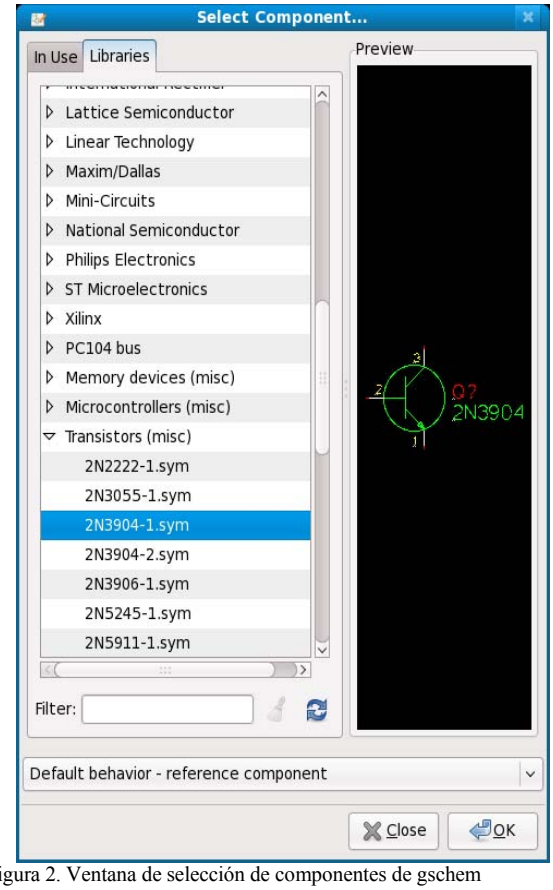


Figura 2. Ventana de selección de componentes de gschem

Para generar el archivo de netlist usaremos la herramienta de línea de comandos *gnetlist* que a partir del esquema de un circuito (archivo con extensión .sch) generará el archivo de texto de netlist con la extensión que queramos (normalmente con extensión .net).

La sintaxis general del programa para convertir el esquema en archivo de netlist es escribir la siguiente orden desde la línea de comandos:

gnetlist -g spice-sdb [opciones] circuito1 ... circuitoN

En el manual puede consultarse todas las opciones disponibles. Una de las opciones más interesantes es crear el archivo de salida con un nombre concreto (por defecto crearía el archivo *output.net*). Por ejemplo para generar el archivo de netlist del esquema guardado en el archivo de nombre *circuito1.sch* con el nombre *circuito1.net* la orden sería:

gnetlist -g spice-sdb -o circuito1.net circuito1.sch

Una vez generado el archivo de netlist el circuito está disponible para ser simulado. Sólo falta usar el simulador para obtener los resultados.

En lo que se refiere a la simulación de circuitos las dos herramientas principales: *ngspice* y *gnucap* son herramientas asociadas al proyecto. Ambas herramientas usan la línea de comandos (aunque hay algunas utilidades creadas para no tener que usar la línea de comandos y hacerlo todo de manera gráfica como la herramienta *gspiceui*). La forma general de trabajar con estas herramientas es leer el circuito a través del archivo de netlist y después mediante órdenes pasados a la línea de comandos hacer las simulaciones (escribiendo en la línea de comandos el tipo y parámetros de simulación). Una vez simulado se representan las variables que deseemos con la orden *plot*. En la figura 3 se observa la línea de comandos una vez se ha iniciado el programa ngspice.

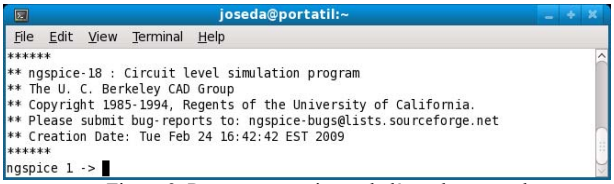


Figura 3. Programa ngspice en la línea de comandos

Para cargar en el programa ngspice un circuito existen dos maneras. Por ejemplo si quisiéramos cargar un circuito cuyo archivo de netlist se llama *ejercicio1.net* se podría hacer de las dos maneras siguientes:

1) ejecutando la orden *ngspice ejercicio1.net* en la línea de comandos.

2) invocando al programa ngspice (figura 3) y posteriormente invocando la orden *source ejercicio1.net* también desde la línea de comandos.

Una vez cargado el circuito ya se puede empezar a realizar las simulaciones. La s simulaciones pueden ser principalmente en continua, en alterna o frecuencia y transitoria. Por ejemplo el comando para ejecutar una simulación transitoria será:

tran paso T_final [T_inicial]

donde *paso* es el incremento que realiza el programa de un punto a otro para realizar los cálculos. *T_final* es el tiempo final hasta el que queremos calcular y *T_inicial* es el primer punto donde queremos que empiece a realizar los cálculos. Si se omite éste tercer parámetro en la orden *tran* el programa lo asumirá como empezar en 0 s. El programa realiza los cálculos y mediante la orden *display* podríamos saber los vectores o variables que el programa puede representar. Para dibujar la gráfica de una variable se usa la orden *plot* mediante la sintaxis:

plot var_1 [var_2] ...

donde *var_1* y *var_2* serían las variables (el programa manipula esas variables como vectores de datos) que queremos representar .

Una manera de automatizar el proceso es modificar el archivo de netlist añadiendo una línea por cada tipo de simulación que deseemos y añadiendo una línea por cada tipo de gráfica que queramos obtener. Cuando el programa lea el archivo de netlist leerá también las órdenes para ejecutar la simulación y realizar las gráficas.

Aunque ngspice y gnucap son herramientas de la línea de comandos que usan una sintaxis parecida existen diferencias entre ellas. La principal es que ngspice una vez realizada la simulación puede representar las ondas de salida directamente y en cambio gnucap requiere el uso de un visor externo de

ondas como puede ser *gwave*. En la figura 4 se obtienen las curvas características de un transistor obtenidas con ngpsice:

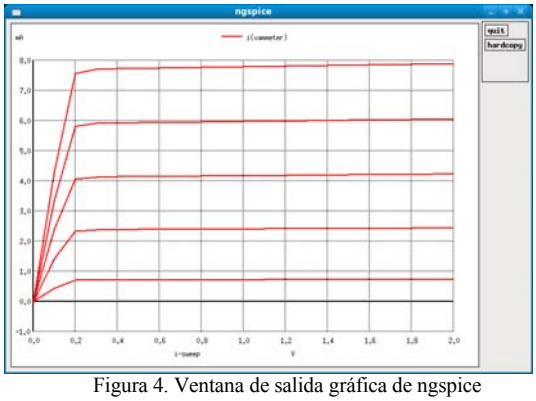


Figura 4. Ventana de salida gráfica de ngspice

Otra diferencia es que con el programa ngspice sólo se puede medir la corriente en las fuentes de tensión y no se pueden incluir probetas de corriente en ningún otro componente. Para solventar este pequeño problema si quisiéramos medir la corriente en un determinado punto del circuito tendríamos que incluir una fuente de tensión de 0 V en ese punto. Así como estas diferencias existen otras entre ambos programas y por tanto es recomendable leer bien los manuales antes de usar uno u otro.

III. QUCS

Qucs (http://qucs.sourceforge.net) es un programa en continuo desarrollo que ha conseguido una interfaz amigable tanto para la captura de esquemas como para la simulación. A diferencia de los programas del proyecto gEDA no requiere el paso intermedio de generar el archivo de netlist, sino que el programa una vez capturado el esquema permite simular directamente añadiendo los bloques de simulación que queramos.

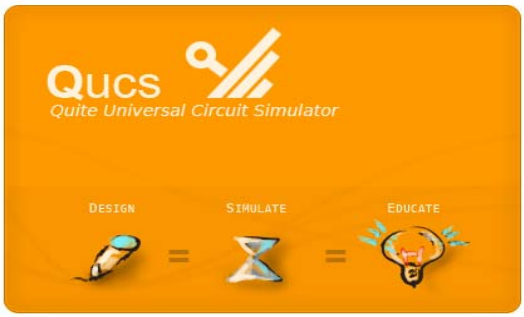


Figura 5. Logo de Qucs

Otra característica más que notable de Qucs es que puede usarse en prácticamente todos los sistemas operativos, es decir puede usarse en sistemas Windows, Linux e incluso Mac.

El programa que está en la versión 0.015 (liberada el 25 de Abril de 2009) tiene muy buenas expectativas de futuro: interfaz amigable, gran cantidad de componentes y posibilidad de añadir nuevos componentes a partir de su modelo, diferentes tipos de simulaciones y diferentes tipos de representaciones gráficas.

El proceso para llegar a simular un circuito con Qucs es muy sencillo. Una vez capturado el esquema se añade el bloque de simulación que deseemos y en dicho bloque están los parámetros de simulación. En la figura 6 se muestran los diferentes tipos de simulación que soporta el programa:

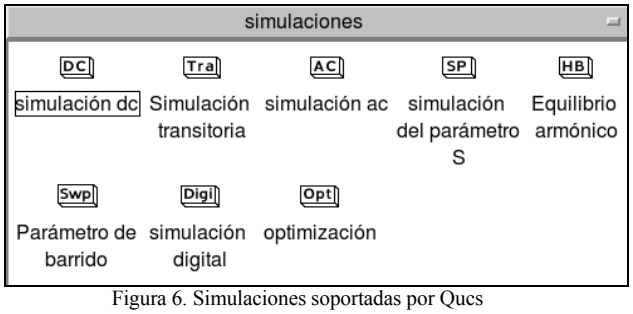


Figura 6. Simulaciones soportadas por Qucs

Una vez simulado el circuito el programa nos abrirá una nueva ventana donde podremos colocar diferentes tipos de gráficas (figura 7), y en cada gráfica seleccionar las variables que queremos representar.

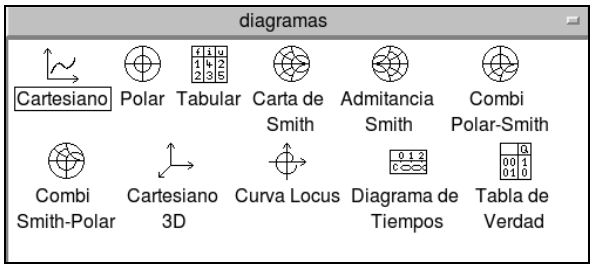


Figura 7. Gráficas que puede realizar Qucs

En las figuras 8 y 9 se muestra el esquema de un circuito con su bloque de simulación y la gráfica obtenida.

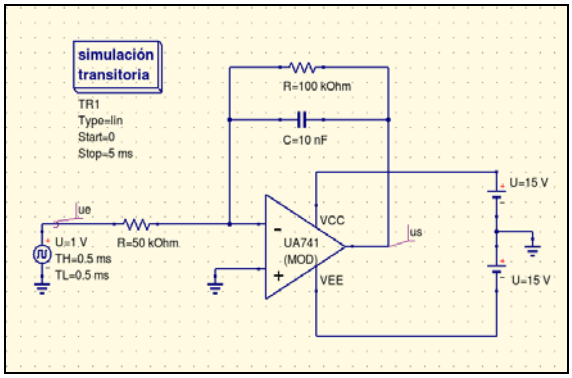


Figura 8. Esquema de un circuito realizado con Qucs

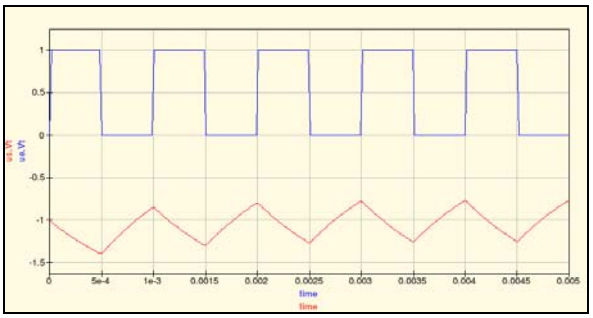


Figura 9. Ventana de salida de resultados del programa Qucs

IV. FEDORA ELECTRONIC LAB (FEL)

Este proyecto (http://chitlesh.fedorapeople.org/FEL/) consiste en un DVD-Live con el sistema operativo GNU/Linux Fedora y multitud de herramientas EDA (no sólo de simulación, sino también de desarrollo de PCBs, VHDL, multitud de librerías, etc.).

Figura 10. Logo de FEL

El DVD está disponible para su descarga en la web del proyecto. Al ser un DVD-Live no instalará nada en el ordenador donde se utilice aunque una vez iniciado habrá un icono en el escritorio por si quisiéramos instalarlo de manera permanente en nuestro ordenador. Las herramientas tratadas anteriormente del proyecto gEDA y el programa Qucs están incluidos también en este proyecto. El listado de las herramientas incluidas en el DVD puede consultarse en la siguiente web:

http://chitlesh.fedorapeople.org/FEL/#Portfolio_Page

V. CONCLUSIONES

La principal conclusión es que existen alternativas potentes en el mundo del software libre para la simulación de circuitos electrónicos. Aunque dado el escaso reconocimiento que tienen hay una gran labor de difusión que realizar. Uno de los ámbitos donde más se puede contribuir con esta labor de difusión es en el ámbito educativo promoviendo o dando a conocer estas herramientas a los alumnos.

REFERENCIAS

[1] PROYECTO gEDA - http://gpleda.org
[2] PROYECTO QUCS - http://qucs.sourceforge.net
[3] PROYECTO FEDORA ELECTRONIC LAB http://chitlesh.fedorapeople.org/FEL/
[4] PROGRAMA NGSPICE - http://ngspice.sourceforge.net
[5] PROGRAMA GNUCAP - http://www.gnu.org/software/gnucap/

Una aproximación a los objetos educativos: eLearning para la Ingeniería Electrónica

Miguel Latorre García
Dep. Ingeniería Eléctrica, Electrónica y de Control, Universidad Nacional de Educación a Distancia
Madrid, España
pelaga@gmail.com

Abstract—**Los recursos educativos abiertos están teniendo un impacto remarcable en la enseñanza a distancia. Sus cambios en la forma de crear y distribuir los cursos suponen una mejora real en la experiencia global de los alumnos e instructores. Dada su mayor interactividad permiten a aquellos interpretar un papel más activo, mientras que a los educadores les facilitan más medios para evaluar todo el progreso.**

Keywords: recursos educativos; objetos de aprendizaje; electrónica; eLearning

I. INTRODUCCIÓN

Desde un principio, la difusión de los microprocesadores en muchos ámbitos de la vida cotidiana estaba destinada a modificar el modo en que generamos los materiales educativos. Los contenidos digitales son un mercado en continua expansión, debido a la amplia diversidad tanto de aplicaciones como formatos (vídeo, texto, imagen) o soportes disponibles donde publicarlos.

Sin embargo, la ausencia de medios con los cuales poder agrupar e intercambiar semejante variedad de elementos dificultaba el acceso a muchas de las actividades desarrolladas en las plataformas de aprendizaje. Cada una de ellas definía su estilo particular para almacenar los cursos. Este hecho ocasionaba en muchas situaciones la incapacidad de encontrarlos, reestructurarlos, así como enviarlos a otros entornos y editarlos posteriormente. Estas islas de información sumadas a la ausencia de alternativas formaban un callejón sin salida que impedía cualquier posibilidad de migración.

A principios de este siglo surgieron varias iniciativas, todas ellas promovidas por diferentes sectores e instituciones de reconocido prestigio, enfocadas a la definición de un formato estándar para la transmisión de contenidos. Los estándares educativos abren las puertas de la creatividad a los desarrolladores de contenidos permitiéndoles despreocuparse de que su obra pueda tener fecha de caducidad por motivos técnicos -por ejemplo, la desaparición de la empresa propietaria del programa con que se creó o la plataforma necesaria para visualizarlo-. Se trata además de evitar la duplicación de esfuerzos fomentando la reutilización de materiales entre las instituciones académicas. Así mismo, incluyen los medios para el reconocimiento de la autoría y las condiciones necesarias para su uso apropiado.

II. CONTENIDOS EDUCATIVOS BASADOS EN OBJETOS

Antes de llegar a los resultados hemos de conocer cuál es nuestro punto de partida. En el método principal que se lleva realizando con gran esfuerzo dentro de los centros de enseñanza, los instructores juegan un rol fundamental preparando diversos materiales como apuntes, referencias bibliográficas y ejercicios relacionados con sus correspondientes asignaturas. Todos estos recursos didácticos tienen en común el uso del papel como soporte, siendo generalmente el ejemplo más característico los libros de texto especializados. Estos recursos se encuentran organizados con un orden creciente de dificultad conforme avanzamos por los temas en que se dividen sus contenidos.

Sin embargo, en la enseñanza a distancia apoyada por el computador e Internet (eLearning) se tratan de aprovechar las capacidades que ofrecen las herramientas informáticas para crear secuencias de actividades que se adapten a la respuesta del estudiante. Está comprobado que utilizando este medio de comunicación se rompen las barreras de espacio y tiempo entre los distintos miembros de la comunidad educativa, pero, hay muchos otros recursos más interesantes tanto para al estudiante como para el instructor. Se busca mejorar el proceso de aprendizaje haciendo que el alumno tenga una postura más activa y, dar al instructor herramientas de evaluación automática de su progreso. Además, éste podrá comprobar cómo sus contenidos alcanzan una audiencia mayor difundiéndolos en la red.

Un movimiento que surgió en el año 2002 está fuertemente vinculado con esta última idea [1]. Los recursos educativos abiertos abarcan desde los contenidos, las herramientas para prepararlos, y los métodos de implementación. En la primera categoría entran los que analizaremos en mayor profundidad, los objetos de aprendizaje (OA en adelante).

Una de estas entidades digitales es una serie de contenidos digitales orientados con un claro propósito educativo [2]. Aquí podríamos encontrar cualquier elemento que pudiésemos utilizar en soporte electrónico: componentes multimedia, documentos, applets, etc. El requisito principal es que tengan un objetivo y los medios para que el receptor pueda asimilarlo. En general, su estructura consta de una breve explicación teórica, un caso práctico de ejemplo, la evaluación final y una descripción acerca de todo el conjunto.

Como se puede observar esta definición es muy amplia, por lo que es motivo frecuente de debate. No obstante, en las múltiples formas que ha presentado desde su origen, se han ido resaltando algunas de sus características más destacadas:

- reutilizable [3],
- carácter abierto [4],
- interoperable, y
- tamaño variable o granularidad.

La interoperabilidad se refiere a la capacidad de mover un objeto entre distintos sistemas de gestión del aprendizaje –o LMS, entornos donde los alumnos y profesores realizan los cursos a distancia- sin pérdida alguna de información. Dicha propiedad fue un gran motivador para la aparición de los estándares en el panorama actual.

En estos momentos también se tienden a integrar los medios para la administración de los contenidos (LCMS), es decir, para su creación y edición. Usando estas aplicaciones remotas el navegador Web se convierte en único requisito imprescindible para los usuarios. Esto supone un ahorro tanto de costes de implantación (no debe adquirir licencias para varios equipos) como de mantenimiento (instalación en un solo equipo). Algunos ejemplos de estos sistemas son ATutor y dotLRN. La Universidad Nacional de Educación a Distancia usa la variante libre ofrecida por este último conocida como OpenACS, ya que permite adaptarla según los intereses de cada institución académica [5].

III. OBJETOS EN LA PRÁCTICA: LOS ESTÁNDARES

Una de las importantes ventajas que ofrece el empleo de los estándares es poder valernos de ellos sin tener que convertirnos en expertos de sus especificaciones. Por ejemplo, no hemos de conocer los entresijos relacionados con los sistemas de transmisión de señal para poder utilizar un televisor.

El modelo de objetos promueve esta filosofía, utilizada en lenguajes de programación como C++ o Java: cualquier contenido se genera mediante otros bloques similares. Un curso, un tema o un simple párrafo de texto son objetos. Lo que los distingue entre sí es su tamaño. Si disponemos de un modo común con el cual estructurarlos y agruparlos seremos capaces –valiéndonos de los programas apropiados– de crear unidades de aprendizaje con tamaño variable mediante unas pocas pulsaciones de ratón.

En la actualidad el modelo de referencia de contenidos compartibles SCORM [6] es el más extendido entre la comunidad educativa para esta tarea. Como tal, es monousuario y éste se gestiona individualmente la experiencia de enseñanza. El diseño del aprendizaje IMS LD es una opción más avanzada, donde los objetos son una pequeña parte de la especificación (añade a más de un participante, los roles de cada uno, o sus relaciones con las actividades y el entorno donde se desarrollan). Por su reducido soporte en las aplicaciones y entornos de aprendizaje sólo lo mencionamos como referencia.

El estándar SCORM es la culminación de aunar los esfuerzos de otros estándares muy conocidos como son XHTML, XML, CSS y Javascript. También denominadas secuencias de aprendizaje o itinerarios, estos recursos están formados por varios archivos vinculados entre sí en páginas Web interactivas. Junto a lo anterior se incluye una descripción del contenido del paquete y otro documento que detalla su estructura: las actividades con sus correspondientes archivos, y, las interacciones entre ellas.

Figura 1. Estructura de una secuencia de aprendizaje cargada en un LMS.

Los metadatos son los responsables de englobar las características más significativas sobre los objetos. El valor de estas descripciones estructuradas es proporcional al nivel de detalle con el cual se elaboren. Por ello en el ámbito de la enseñanza el estándar de facto es IEEE LOM [7], donde un documento XML contiene separado en sus nueve categorías una serie de elementos. Su extensión es lo suficientemente elevada y al ser todos sus componentes opcionales, cubre la mayor cantidad de aspectos de un recurso. Entre ellos se encuentran elementos genéricos como el título o la fecha de creación, los requisitos educativos (edad, conocimientos previos) y técnicos, pero también los autores que han contribuido a su elaboración, la licencia de uso, su nivel de agregación, las relaciones con otros recursos y su clasificación.

Para hacernos una idea del potencial que ofrecen los metadatos descriptivos no hemos más que acudir a un comercio. Cuando vamos a adquirir un producto y hemos de abonar su coste, el comerciante no tarda más que unos breves instantes en conocer su precio. Ni siquiera ha de comprobar a qué modelo o marca pertenece porque para determinarlo sólo ha de leer su código. Análogamente, los metadatos nos permiten averiguar la información guardada en los documentos digitales a los cuales van asociados sin necesidad de abrirlos. Son por este motivo imprescindibles para la reutilización de los objetos educativos en su totalidad, aunque lo ideal sería no tener que manipularlos directamente [8].

Los documentos estructurados en XML incrementan considerablemente el rango de los formatos de salida. Si efectuamos la conversión un mismo recurso y la información contenida en él será adaptable a diferentes lectores. Una muestra de ello sería traducir la descripción a diferentes idiomas, a partir del texto escrito por el autor original, en el momento de su visualización –una sola fuente para varios destinatarios–. Todavía aumentaríamos más su valor

asegurando su completa accesibilidad. Un caso particular cercano a electrónica lo tenemos en las descripciones de nodos de un circuito electrónico (archivos ascii SPICE). A través de un proceso de conversión lo exportaríamos a gráfico vectorial SVG y obtendríamos una imagen del circuito, o bien a un archivo de sonido con su explicación paso a paso, e incluso al código de lectura y escritura táctil Braille.

Indicar cuál es la licencia de uso más adecuada para el creador de contenidos no es una cuestión menor, pero este factor ni siquiera aparecía antes de los objetos de aprendizaje o las publicaciones en páginas Web. Identificar al responsable de la creación de un determinado recurso educativo permite también asegurarnos de su calidad, ya que ha pasado unos controles estrictos hasta llegar a su publicación. Además aporta una valoración del trabajo anterior, al permitir citar las fuentes en las que se ha basado para su desarrollo. Esta colaboración a distancia puede ayudar en aquellos casos donde obtener las fuentes bibliográficas originales no resulte una tarea sencilla. Es por ello que aquellas licencias libres como la Creative Commons By se están adjuntando a los recursos educativos, pues otorgan a los usuarios la seguridad de poder utilizarlos de mutuo acuerdo con el reconocimiento de sus autores.

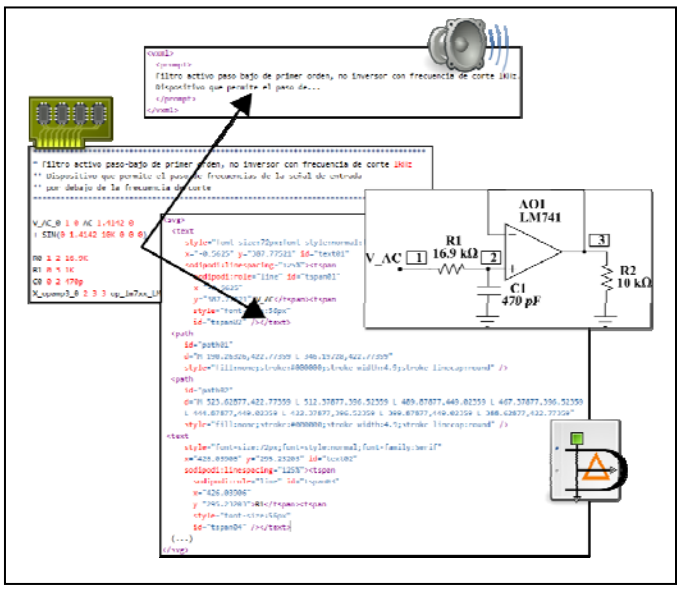


Figura 2. Conversión de recursos a distintos formatos

Como se ha resaltado antes, por una parte tenemos la organización del objeto o el orden de las actividades que lo forman (sus secciones: introducción, desarrollo y evaluación) y, completamente independiente a la misma, los archivos asignados a cada una de ellas. Hay una clara separación entre los contenidos físicos y cómo deben aparecer posteriormente en el navegador. ¿Qué ventajas nos aporta el uso de este criterio?

Mediante la incorporación de varias organizaciones en un mismo objeto podemos utilizarlo en múltiples contextos. Por ejemplo, supongamos un recurso que trate sobre el proceso de fabricación de placas de circuito impreso. En una organización inicial se podrían indicar las distintas etapas, actuando así de índice de contenidos para el instructor en su exposición en la clase. Sin embargo, otra opción sería crear un árbol de estados que controlase cómo ir de una etapa a otra y comprobar antes del laboratorio si el alumno lo realiza correctamente. Este salto abre muchas posibilidades, desde simulaciones de procesos, procedimientos, a árboles de actividades que tuviesen caminos variables según las respuestas del estudiante. Avanzamos de un esquema lineal a enlaces conectados entre sí con varios caminos posibles.

Aunque parezca extraño, esta funcionalidad se incrustaba en los primeros gestores de aprendizaje, resultando complicada su modificación con otros medios externos.

El último elemento que falta por mencionar es el motor que controla el movimiento entre las distintas actividades y notifica de su superación: el entorno de ejecución. Su función es controlar la comunicación de las acciones del estudiante: avance y retroceso de página, duración máxima de cada actividad, comprobación de verdadero/falso, mensajes de aviso, etc. Esta información se envía al LMS para generar registros de estadísticas de forma automática con su progreso. Destacar esta funcionalidad, pues sin intervención alguna por parte del instructor en este proceso -aún más costoso conforme aumentase el número de participantes- le permite consultar su evolución en cualquier momento.

IV. NUEVAS HERRAMIENTAS PARA E-LEARNING

Hemos visto en los apartados anteriores qué es un OA, su formato en la práctica y sus características más relevantes. Igual de importante es conocer cómo crearlo pero, más aún, encontrar otros objetos e interrelacionarlos. En esta sección se resume dicha tarea, aportando métodos para convertir trabajos previos como documentos, presentaciones, o videos y métodos que agilicen la autoría de objetos desde cero.

Si hay una constante que se repite en la infraestructura para crear objetos es la existencia de una estructura bien definida. El procedimiento para crear un OA es similar al empleado en la edición de documentos con procesadores de texto. Primero nos preparamos una plantilla típica: encabezado, cuerpo y pie de página, formato de títulos y viñetas, etc. Luego insertamos los contenidos separándolos por secciones según un orden preestablecido: los ejercicios aparecen en una sola página/carpeta, las preguntas de la evaluación se ponen cada una en páginas diferentes...

La diferencia fundamental es que esa organización está separada por completo de nuestros archivos y el autor programa las interacciones con una herramienta especializada como el editor Reload. En ella se incluyen eventos como las condiciones para avanzar de una actividad a otra, es decir, la puntuación, el nº máximo de errores y similares. Es evidente que para visualizar y empaquetar el recurso SCORM es necesario cargarlo en un reproductor SCORM o un LMS.

Crear recursos interactivos de calidad es laborioso, por ello parece una consecuencia razonable procurar adaptar materiales que hayamos elaborado con anterioridad con aportaciones de otros autores. Los repositorios de objetos educativos son los responsables de facilitar la localización de estas últimas.

Cuando nuestra meta sea encontrar recursos didácticos sobre un tema muy específico los mejores sitios a los que se puede acudir son los repositorios institucionales. Los buscadores están orientados a encontrar información, pero, interesa conseguir documentos y resultan poco eficaces en ello. La tendencia es que pasemos de ser buceadores de largas listas de posibles coincidencias en un buscador, a exploradores que descubren su idea inicial en mente. Esto implica que nuevamente las tecnologías se adapten a nuestra forma de pensar y no viceversa.

Un repositorio es un servicio más en la red académica para almacenar los metadatos o los mismos con los recursos que describen. En la UNED nos referimos al e-spacio [9]. Para los estudiantes e instructores actúa como una herramienta más que les sirve para explorar contenidos, descargarlos a su equipo local o enviarlos directamente a su gestor de contenidos. Al acceder a cualquiera de los existentes en la red de redes, dispondremos de un árbol desde el cual filtrarlos según su disciplina (matemáticas, física...), tema, departamento, dificultad y muchos otros parámetros. Este avance es notable porque pasamos de una navegación jerárquica con información muy limitada –típica en un explorador de archivos– a un modelo basado en categorías. Éstas se relacionan mediante ontologías, las cuales se generan con los contenidos presentes en los metadatos. Mediante ellas se restringen o amplían el abanico de resultados sin nuestra participación.

Queda pendiente un punto a estudiar: la generación de objetos con ficheros no estructurados. Se verá con un ejemplo concreto. El departamento de Ingeniería Eléctrica, Electrónica y Control (DIEEC) disponía de una amplia recopilación de ejercicios, simulaciones y tutoriales relacionados con las asignaturas impartidas en los cursos de Electrónica. Dada la elevada cantidad de archivos no era viable editar los metadatos uno a uno.

Un análisis de los libros de texto a los cuales acompañaban, combinado con ciertos campos que se extrajeron con funciones propias del sistema operativo (aspectos objetivos: fecha, tipo, espacio ocupado) permitió describirlos de una forma semiautomática [10]. Con un simple lenguaje de script se generan los archivos estructurados en el menor tiempo posible. Primero se organizaron los materiales de acuerdo a criterios como asociar sus carpetas con las categorías arriba mencionadas. Los nombres de archivo sirvieron como identificadores únicos para definir unívocamente las trayectorias de acceso desde el navegador. Cruzando esos detalles con los restantes de los textos originales se obtuvieron las instancias XML, validándolas frente al esquema LOM para comprobar su conformidad con el estándar. Empaquetar el conjunto recursos+metadatos con los medios de evaluación y su estructura es inmediato una vez se encuentran en el repositorio. Ahora si observamos un objeto en concreto, éste tiene asignado un documento con su descripción para que desde cualquier repositorio se pueda importar en otras plataformas de aprendizaje.

V. CONCLUSIONES

Un mantra de la ingeniería afirma que cuando nos enfrentamos a un problema en muchas ocasiones "hay más de una forma de solucionarlo". El eLearning se caracteriza por ofrecer muchas fuentes de consulta al estudiante, para ayudarlo en la difícil tarea de transformar la información sobre la materia que le interesa en conocimiento.

Obtener informes de seguimiento, estadísticas o gráficas de los recursos más consultados o generar resúmenes de todos los contenidos son buenas justificaciones para fomentar su uso, pero no nos podemos olvidar algunas de sus limitaciones. Una de las metas es conseguir el equilibrio entre la accesibilidad de las actividades frente a su interactividad. También hemos de abarcar a la mayor audiencia posible, independientemente de sus aptitudes.

Los estándares educativos no sólo permiten disponer de una gran lista de recursos que luego se combinan en ejercicios o bloques mayores, o dividir unidades completas a nuestro antojo para luego reutilizarlas. Estos materiales proceden de trabajos previos realizados por otros autores e implican una adecuada integración con el objeto de enriquecer los cursos ya existentes.

REFERENCIAS

[1] D. Wiley, “On the sustainability of open educational resource initiatives in Higher Education” [informe en línea]. OECD. 2006. http://opencontent.org/docs/oecd-report-wileyfall-2006.pdf

[2] A. Chiappe. “Definición de Learning Objects”, URL con último acceso el 5/07/2009. http://andreschiappe.blogspot.com/2007/09/que-es-un-objeto-de-aprendizaje-what-is.html#links.

[3] T. Boyle, R. Windle, D. Leeder, H. Wharrard, R. Alton, and J. Cook, (2006). “An Agile Method for Developing Learning Objects”. Full paper, ASCILITE, Sidney, Australia, 3-6 December, 2006.

[4] G. Fulantelli, M. Gentile, D. Taibi, and M. Allegra, (2008). “The Open Learning Object model to promote Open Educational Resources”. Journal of Interactive Media in Education. http://jime.open.ac.uk/2008/09/.

[5] E. Sancristobal, S. Martin, R. Gil. G. Díaz, A. Colmenar, M. Castro, J. Peire, J.M. Gómez, E. López, P. López, “Integration of Internet Based Labs and Open Source LMS”, iciw, pp.217-222, 2008 Third International Conference on Internet and Web Applications and Services, 2008

[6] Advanced Distributed Learning (ADL). 2004. SCORM 2004 3th Edition. Sharable Content Object Reference Model Download. Consultada en Junio del 2009: http://www.adlnet.gov/downloads/DownloadPage.aspx?ID=237

[7] IEEE Standard for Learning Object Metadata. ANSI/IEEE. Sitio web: http://ltsc.ieee.org/wg12/(2002)

[8] E. Duval and W. Hodgins, “Making Metadata go away – Hiding everything but the benefits”, International Conference on Dublin Core and Metadata Applications (DC-2004), 2004.M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

[9] A C. Lagoze, S. Payette, et al. “Fedora: an architecture for complex objects and their relationships”, International Journal on Digital Libraries, Vol. V6, No. 2. (April 2006), pp. 124-138

[10] K. Cardinaels, M. Meire and E. Duval, “Automating metadata generation: the simple indexing interface”. Proc. of the 14th international conference on World Wide Web, pp. 548-556, 2005.

Proceso automático de formación de archivos de metadatos

Manuel Blázquez Merino
Ingeniero Técnico Industrial
Proyectante en Departamento de Ingeniería Eléctrica, Electrónica y de Control - UNED
Madrid, España
manuel.blazquez.merino@gmail.com

Pablo Blázquez Merino
Ingeniero Superior en Informática
Programador Senior
Nimbus Systems S.L.
Madrid, España
pablo_blazq@yahoo.es

Abstract— **El presente escrito presenta un procedimiento abierto de conversión de metadatos en ficheros XML. Para ello, una vez que los metadatos hayan sido recogidos en una base de datos, se puede lanzar un proceso de lectura y recuperación de los metadatos como elementos componentes de un registro y tratarlos como etiquetas, a fin de integrarlos en un fichero XML. El procedimiento que se presenta no tiene una limitación determinada en el número de archivos a generar y es fácil de adaptar para cualquier estructura de base de datos. Los ficheros XML generados son compatibles con los estándares actuales de forma que se pueden integrar en motores de búsqueda, plataformas o repositorios de documentación o elementos hipermedia.**

Keywords: metadata; XML; IEEE-LOM; file generation; reusable object

I. INTRODUCCIÓN

El reconocimiento de una obra desarrollada en cualquier tipo de medio utiliza principalmente los metadatos extraídos de ella misma para su divulgación. Dependiendo del uso y sobre todo, del tipo de medio que se utilice para su divulgación, el número de metadatos útiles es variable. Por ejemplo, para una obra literaria, los metadatos que se pueden utilizar pueden ser pocos, quizá una decena, dado que hablamos de una obra aislada en la que, para su divulgación se han de destacar datos referentes a su edición, su autoría y su contenido. Cuando se pretende ampliar el uso de metadatos definitorios a una colección de obras, el número de metadatos se amplía con objeto de definir el contexto de la colección.

Por supuesto, los contextos son igualmente variables, dado que prácticamente, cualquier colección puede integrarse en un conjunto mayor. Para concretar estos contextos con un ejemplo de fácil comprensión y muy familiar para el tipo de lector de artículos como el presente, puede imaginarse un proceso de recuperación de datos de todas las obras que componen las sesiones de un congreso o de un conjunto de congresos afines entre sí. En este caso el conjunto de metadatos definitorios se han de extender a la obra en sí, a la sesión a la que pertenecen y al congreso que los acoge. En suma, la definición de un objeto reutilizable dependerá de él mismo y de su contexto.

II. LOS FICHEROS DE DEFINICIÓN DE METADATOS

Entre los múltiples lenguajes que se pueden utilizar para acoger los metadatos, el más sencillo y flexible para un uso amplio es XML [2]. Detrás de estas siglas se desarrolla el lenguaje denominado *eXtensible Markup Language*, es decir, un lenguaje de marcado extensible. Los lenguajes de marcado han estado presentes desde el principio de los sistemas operativos, dado que era el medio de marcado de las configuraciones, lo que comúnmente se conoce como archivos INI.

Con el auge de Internet y con la red, del lenguaje HTML se han desarrollado múltiples lenguajes que permiten la distinción entre datos y metadatos, habiendo seguido todos ellos un formato prácticamente común.

Para concretar la utilidad de los archivos XML en la definición de objetos reutilizables, se puede decir que entre ellos tiene que haber una referencia univoca, es decir, sea cual sea la estructura de archivos que se pretenda formar, el archivo XML asociado a un objeto, ha de disponer de su dirección o ubicación además de los metadatos de referencia.

El trabajo que ha precedido a este documento conlleva, entre los objetos y sus ficheros XML definitorios, un tercer componente: una base de datos que acoge inicialmente los metadatos mediante tablas relacionales. El hecho de elegir una base de datos de este tipo reside en los servicios añadidos que ofrecen por su facilidad de uso, por la posibilidad de completar información mediante la ejecución de consultas y por la disponibilidad de exportación en formato de hoja de cálculo mediante el cual se pueden realizar estudios estadísticos relacionados con el contexto de las obras que se acogen.

III. LA DEFINICIÓN DE LAS ESTRUCTURAS DE METADATOS

El proceso presentado en este documento comienza con un repaso exhaustivo de los objetos reutilizables en orden a concretar cuales son los metadatos que han de utilizarse para definir un objeto con suficiente precisión. Esto conlleva la realización de una estructura de tipo árbol que sostenga de una forma coherente los metadatos. Lógicamente, existen infinitos modelos de estructuras. Este es el momento de revisar los estándares existentes.

Entre los más extendidos, dado su carácter relacionado con el ordenamiento y organización en bibliotecas y colecciones de libros es Dublin Core. De hecho, de los más extendidos es quizá el que más tiempo lleva funcionando. La iniciativa DCMI (*Dublin Core Metadata Initiative*) [3] ha aplicando una metodología sencilla basada en la existencia de dos niveles, el nivel *Simple* y el nivel *Qualified*. El primero se compone de 15 elementos y el segundo incluye un conjunto de elementos cualitativos o "adjetivos" que profundizan en la concreción del metadato a contener. Estos últimos siguen el siguen el principio de mutismo, es decir, los elementos de nivel simple pueden existir sin necesidad de calificadores. De esta forma, se puede enfocar la estructura del fichero XML en diversos grados de profundización en la definición del objeto reutilizable.

Pero el estándar que logra un mayor grado de definición es LOM (Learning Object Metadata) [1] desarrollado por IEEE. Su estructura amplia permite abarcar objetos que requieran gran cantidad de metadatos. El objetivo de IEEE con este estándar reside en potenciar la divulgación de los objetos que ellos mismos definen como Objetos de Aprendizaje. De hecho, el estándar no solo define la estructura de metadatos sino que además propone un unidad definida como SCORM [5], [6] que agrupa al objeto reutilizable, al fichero de metadatos LOM y un manifiesto ad-hoc. Estas unidades SCORM son fácilmente transportables, dada su condición de fichero comprimido, y son aceptadas para su carga por la mayoría de los repositorios institucionales.

LOM define un agrupamiento de metadatos basado en una disposición jerárquica de nueve grandes grupos: [*General, LifeCycle, Meta-metadata, Technical, Educational, Rights, Relation, Annotation, Classification*], que soportan conjuntamente hasta 64 metadatos. A partir de esta definición, siguiendo las recomendaciones de uso de los campos de marcado, LOM permite la definición de prácticamente cualquier tipo de obra. Precisamente, el número de metadatos que el autor pretende usar para la definición del objeto queda a su demanda, por lo que en multitud de ocasiones bastará con una veintena de metadatos para cubrir suficientemente las necesidades definitorias.

IV. Creación de la base de datos: segundo paso en el proceso

Dado que LOM define suficientemente los metadatos a utilizar, el siguiente paso consistirá en crear una base de datos que acoja la estructura de metadatos. Una buena práctica en la manipulación de documentación, consiste en utilizar un código específico (e.g., DocCode) que, de forma unívoca, sirva de etiqueta de referencia del documento completo. En este caso, se pueden utilizar múltiples ontologías de clasificación, como el método decimal utilizado en biblioteconomía [7], el sistema (LC) L*ibrary of Congress* [4], sistemas *NATO*, o cualquier medio de codificación industrial de documentación.

Este será el primer campo de la tabla que acogerá los metadatos y que será definido como clave principal de la tabla, cuya característica fundamental es la no duplicación de datos en su seno. El resto de campos tendrán como nombre, la misma expresión que LOM define para el metadato asociado.

Dado que una base de datos contiene tablas relacionadas, se pueden generar tantas como sea preciso para adaptarse al contexto. Siguiendo el ejemplo de los congresos citado al principio de este documento, su estructura de tablas podría componerse de una tabla "Congresos" en la que se guarde la información genérica de un congreso dado (año, ciudad, organismo organizador, etc.), una segunda tabla que identifique los tipos de sesiones que habitualmente los comités organizadores proponen de forma temática, una tercera tabla que contenga los metadatos de un objeto tipo ponencia, y diversas tablas que contengan códigos de materias o temáticas, códigos de ciudades, etc.

V. El proceso automático de generación de ficheros XML

Una vez que se han definido las tablas que contienen los metadatos y las consultas que relacionan dichas tablas y por tanto, generan la estructura que acogerán los ficheros XML, bastará iniciar un proceso automático de generación de ficheros XML.

El proceso que se ha desarrollado a lo largo del trabajo dispone de un objeto fuente (la base de datos), un programa lanzadera (metagenerator) y un fichero destino (plantilla LOM), que se irá clonando de forma sucesiva tantas veces como archivos XML sea necesario generar.

En la figura 1 se observan los componentes de la aplicación, la cual dispone de un fichero de configuración y la propia base de datos relacional como elementos de entrada y la plantilla de metadatos como elemento de salida.

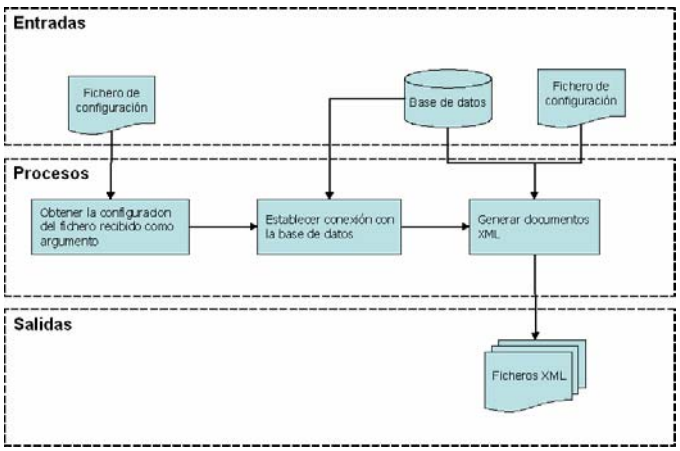


Figura 3. Esquema de entradas, proceso y salidas del programa lanzadera

El fichero de configuración es el argumento de entrada del programa, y contiene información que la aplicación necesita para su funcionamiento. Este fichero contiene un formato formado por una serie de líneas en las que se indica una propiedad y su valor mediante la forma:

```
<propiedad>=<valor>
```

En su estructura se han determinado tres parámetros configurables: el acceso a base de datos, la plantilla XML y directorio de salida.

El primer parámetro de acceso a base de datos, a su vez dispone de la definición de cuatro propiedades. La primera de ellas *db.driver,* se trata de la definición de las clases de Java encargadas de establecer la conexión y realizar las operaciones en base de datos. Estas clases Java está ya desarrolladas, bien al estar presente en la propia distribución de Java o bien al poder obtenerse de la compañía a la que pertenece la base de datos. En la presente aplicación, se ha decidido la posibilidad de determinar valores para el acceso y operativa sobre bases de datos de uso generalizado. En concreto, se ha determinado configuraciones para bases de datos Microsoft Access, bases de datos de Oracle y de SQL Server. A continuación se muestra el código de configuración para cada una de ellas respectivamente.

```
db.driver=sun.jdbc.odbc.JdbcOdbcDri
ver
db.driver=oracle.jdbc.driver.Oracle
Driver
db.driver=com.microsoft.jdbc.sqlser
ver.SQLServerDriver
```

La segunda propiedad *db.url* es indicativa de la cadena de conexión a base de datos. Para conectar a una base de datos de Microsoft Access, se utilizaría la expresión:

```
db.url=jdbc:odbc:MS Access
Database;DBQ=<ruta del archivo mdb>
db.url=jdbc:odbc:MS Access
Database;DBQ=config/[base de
datos].mdb
```

En cambio para conectar a una base de datos en Oracle, se utilizaría:

```
db.url=jdbc:oracle:thin:@<host>:<pu
erto>:<Base de datos>
db.url=jdbc:oracle:thin:@localhost:
1521: <Base de datos>
```

Finalmente, la expresión elegida para la conexión a una base de datos en SQL Server, se utilizaría:

```
db.url=jdbc:microsoft:sqlserver://<
host>:<puerto>;DatabaseName=<nombre
_BD>
```

La tercera propiedad *db.username,* indica el usuario de base de datos. En caso de no existir configuración de usuario quedará vacío, pero si se configura se utiliza la expresión:

```
db.username=test
```

Finalmente el cuarto y último parámetro de este bloque de configuraciones, *db.password*: pretende introducir un elemento de seguridad mediante la indicación de una contraseña de base de datos. El valor quedará vacío si no se desea activar esta opción. La expresión utilizada será:

```
db.password=test
```

El segundo bloque de configuraciones, las de la plantilla de XML, se indica mediante una única propiedad *xml.template*, mediante la cual se indica la ruta que servirá como plantilla para generar todos los ficheros XML. Así la propiedad se indicará mediante:

```
xml.template=config/plantilla_LOM.x
ml
```

En último término, se ha de indicar la ruta del directorio de salida de los ficheros XML que se vayan generando. Esto se lleva a cabo mediante una única propiedad *xml.output.directory*. y su expresión queda indicada del siguiente modo:

```
xml.output.directory=output_LOM
```

En definitiva mediante el fichero de configuración quedan definidas las conexiones entre entradas y salidas a las que alude el esquema de la figura 1.

Por otra parte queda por definir como se lleva a cabo el proceso generador. En esencia, tal y como se reproduce en el algoritmo de la figura 2, el programa lanzadera se encargará de recorrer todos y cada uno de los registros de la tabla origen, copiar su contenido y volcarlo en la etiqueta del fichero de destino. En realidad se trata de un programa recursivo que extrae tanto datos de la base como existan, con lo que la función de coste del programa es lineal y dependerá del número de registros se quieran reproducir en ficheros XML.

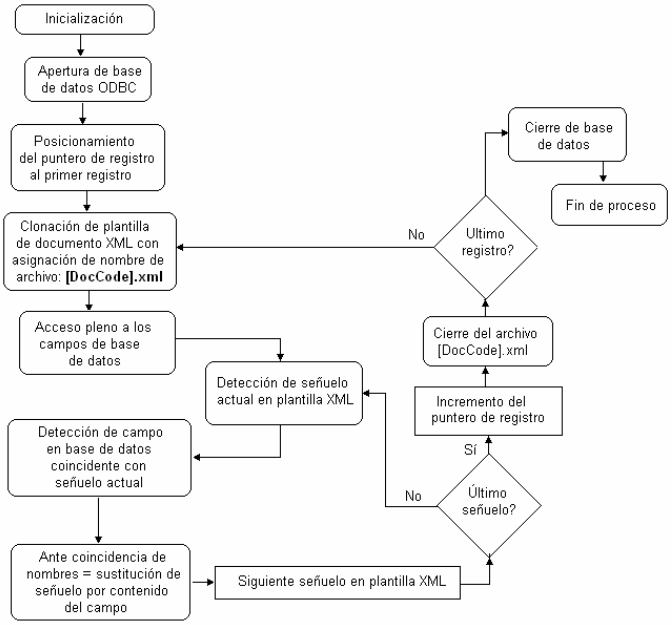


Figura 4. Algoritmo gráfico del programa lanzadera

VI. LA PLANTILLA XML Y SUS SEÑUELOS

Como se ha indicado con anterioridad, tanto la base de datos relacional como el proceso de la aplicación, están al servicio del elemento que sirve de vehículo para la generación de los ficheros XML: la plantilla de metadatos LOM. La estructura de esta plantilla contendrá elementos que serán sustituidos cuando se convierta en documento XML. Para entender su estructura se ha mostrado un aspecto del código de la plantilla XML en la figura 3 mediante bloques marcados.

El primer bloque marcado consistirá en la indicación del lugar del que se han de leer los datos. En el caso del ejemplo, se trata de una tabla denominada "Principal" , la cual está indexada por medio de un campo clave que además será el que dé lugar al nombre del archivo XML a reproducir. En ella, se observa una primera declaración de los campos que servirán de señuelo. El programa se encargará de encontrar cada uno de ellos en profundidad.

Un aspecto crítico de esta solución es que tanto en la base de datos como en la declaración inicial en la plantilla ha de existir una absoluta coincidencia de nombres. De no existir coincidencia, el programa no se ejecuta y emite un mensaje de error, quedándose el proceso bloqueado. En un tipo de código textual como XML ha de llevarse a cabo una cuidadosa labor de escritura y comprobación de nombres, tanto en la declaración como en los posteriores señuelos. Respecto a los señuelos, se ha decido que se den a conocer mediante el carácter precedente "$".

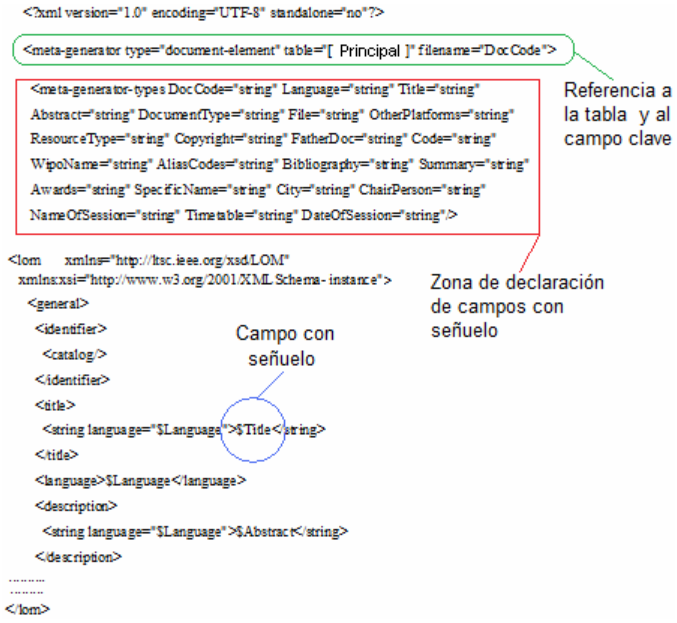


Figura 5. Código fuente XML – Inicio de archivo

VII. IMPLEMENTACIONES FINALES EN EL PROGRAMA LANZADERA

Una de las características de los campos de marcado de metadatos en LOM es que IEEE no recomienda la utilización de conjuntos múltiples de datos en un mismo metadato. En un fichero de metadatos, el nombre del autor es esencial para su definición y divulgación. Pero, ¿qué ocurre en el caso de que existan varios autores firmando una obra? IEEE-LOM determina que pueda existir multiplicidad en etiquetas, es decir, en el caso del ejemplo, debería haber tantas etiquetas de metadatos de autor como autores haya.

En este caso, hay que mejorar y adaptar el proceso descrito en el capítulo V, dado que tan sólo se dispone de una tabla en la que se acumulan todos los metadatos. El equipo encargado de recoger los metadatos del origen e incluirlos en su campo y registro correspondiente en la base de datos, tendrá que modificar esta situación.

La solución más efectiva es construir y disponer de tantas tablas adicionales como campos con datos múltiples existan. Utilizando el ejemplo anterior, deberá de existir una tabla Principal de metadatos y una tabla de autores. La conexión entre ellas se llevará a cabo por coincidencia en un campo común. Este campo es el que se ha recomendado en el capítulo IV definido como DocCode o código específico y unívoco para cada registro.

En la figura 4 se propone un ejemplo que sigue esta organización. Así, existirá una tabla "Principal" que contiene los campos que definen un objeto reutilizable, encabezados e indexados por el campo DocCode o Código específico de documento, ya comentado en el apartado anterior, que sirve de nombre para los archivos XML que se produzcan.

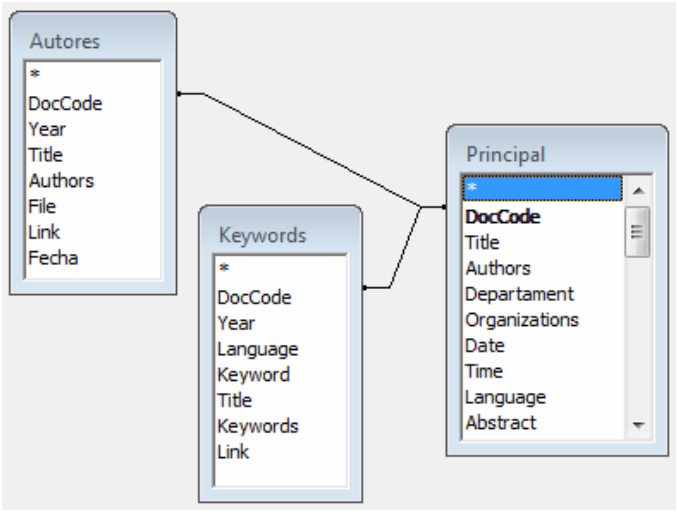


Figura 6. Ejemplo de solución para los campos múltiples. Relación de tablas en la base de origen de datos

Relacionado a través de este campo, existe un listado de registros en los que principalmente se dispone de un código de documento y un autor. Así podrán existir múltiples registros con el mismo DocCode si en el objeto original existen múltiples autores que firman la obra.

De igual manera, según se vayan incorporando nuevas tablas, todas tendrán la misma referencia al campo DocCode, con lo que la figura gráfica de relaciones entre tablas de la base de datos en conjunto se convertirá en una estrella, con la tabla principal como nodo central. Tan solo queda solucionar como se implementa esta estructura en la plantilla XML, que hasta este punto sólo tiene acceso a la tabla principal.

En la figura 5 se ha detallado parte del código fuente, donde se puede observar una estructura repetitiva para dos tipos de datos que habitualmente son múltiples, los campos de autores y los campos de keywords o palabras clave.

En la primera de ellas, se puede observar que la acción "meta-generator" declara un bloque separado en la que la información y señuelos que existan en su interior, se obtendrán de la tabla "*Keywords*", siendo el campo "*DocCode*" el que permitirá su volcado. Se tendrá que indicar posteriormente todos aquellos campos que sostiene la tabla en cuestión, aunque la definición de todos ellos es opcional. Tan sólo es necesario indicar, como mínimo, aquellos que queramos que se encuentren en la estructura repetitiva. En el primer bloque marcado, se utilizarán tanto el campo "*Language*", como el campo "*Keyword*", que indica en el idioma se ha escrito la palabra clave.

En el bloque inferior marcado en la figura 4, se muestra una estructura mayor, que hace referencia a la tabla "*Autores*", también indexados por el campo clave "*DocCode*", en referencia directa al ejemplo de la figura 3. En este ejemplo, se mezclan etiquetas de tipo fijo y etiquetas múltiples dotadas de señuelo. En este caso, cuando la plantilla se convierta en documento XML aparecerán bloques consecutivos entre ambas etiquetas <contribute>…..</contribute>

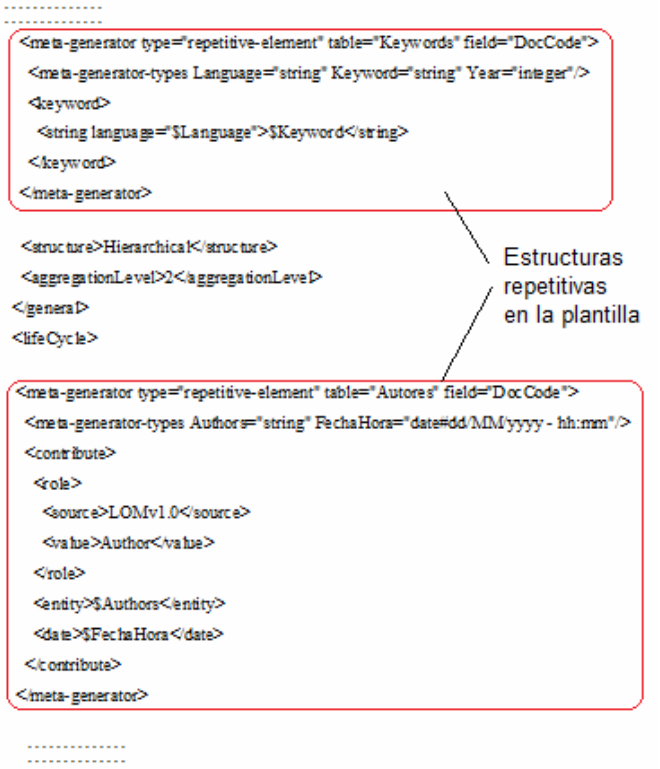


Figura 7. Detalle de código fuente XML para la reproducción de estructuras de datos repetitivas

VIII. PRUEBAS EN LA EJECUCIÓN DEL PROGRAMA LANZADERA

Una vez, que se ha dispuesto de toda la estructura de base de datos con los datos disponibles y la plantilla adecuadamente estructurada, se puede ejecutar el programa lanzadera. En primera instancia se ha probado el presente programa para la ejecución de los metadatos recogidos de las ponencias publicadas durante ocho congresos del proyecto TAEE (Tecnologías Aplicadas a la Enseñanza de la Electrónica). Cada ponencia consta de 18 metadatos de tipo simple y 7 estructuras repetitivas, que insertan aproximadamente 25 metadatos múltiples en cada documento XML. Esto da una cifra media de 43 inserciones por documento. En diversas pruebas efectuadas bajo estas condiciones, se ha obtenido un tiempo medio de ejecución para una cantidad de 964 ponencias de 2 minutos y 2 segundos, lo que significa que se ha creado un documento XML en un tiempo medio de 130 milisegundos.

Se ha llevado a cabo un segundo set de pruebas de datos masivos extraídos de los componentes de cada documento, en concreto objetos como gráficos, diagramas, tablas, fotografías, web snapshots, etc. En este caso, el programa lanzadera tenía que generar metadatos para 4431 ficheros. Estos 4431 archivos XML tienen una estructura similar a la del primer set de prueba. En este caso el tiempo de ejecución medio del set completo fue de 7 minutos y 55 segundos, lo que da un tiempo medio de 107 milisegundos por fichero XML.

Se puede observar una desviación en el valor medio del tiempo entre ambos sets de pruebas que pueden achacarse fundamentalmente a la existencia de un primer gasto en tiempo

de inicialización de valor constante sea cual sea el volumen de datos a procesar. Otra de las circunstancias que motivan esta desviación es atribuible a los procesos ajenos al programa en ejecución por parte del sistema operativo del equipo de prueba o situaciones aleatorias de interrupciones del sistema o atención a tareas propias del equipo.

Para obtener un tiempo estimado de respuesta del programa se ha decidido linealizar los datos de ambos sets de prueba, dividiendo el tiempo total de proceso en dos partes: un primer tiempo de retardo por inicialización de tipo constante y un segundo tiempo proporcional al número de documentos a procesar. Así, mediante el citado cálculo se han obtenido un tiempo de retardo de 23, 85 segundos y un tiempo medio estimado de 101,8 milisegundos por fichero XML procesado.

Por supuesto, esta cifra es mejorable, sobre todo si se pretende utilizar el programa sobre datos masivos con un orden superior a 10.000 documentos procesados. La reducción de los tiempos de procesado es un objetivo a corto plazo que se marcan los autores.

No obstante, esto no minimiza la verdadera ventaja de la aplicación presentada, dado que el programa lanzadera es posible adaptarlo con bastante sencillez a cualquier proceso de creación de documentos de metadatos, sea cual sea su procedencia. Tan solo es necesario acogerlos en una base de datos y adaptar su estructura en una plantilla como la citada en este documento.

IX. CONCLUSIONES

A lo largo de esta publicación que ha querido mostrar un software abierto y de fácil adaptación a cualquier proceso de recuperación y tratamiento de metadatos. Del trabajo desarrollado ha resultado un algoritmo de ejecución rápida que conserva todas las características exigibles a un documento XML según el estándar IEE-LOM. Con posteriores adaptaciones es posible que siga otros estándares, generando los documentos de la misma forma.

AGRADECIMIENTOS

Los autores quieren agradecer al Ministerio de Ciencia e Innovación de España y al Plan Nacional Español I+D+I 2008-2011 el apoyo a este artículo dentro del proyecto RedOBER - Proyecto TSI2007-31091-E Objetos Educativos Reutilizables (para el EEES en las especialidades de las Tecnologías de la Información y las Comunicaciones).

REFERENCIAS

[1] Learning Technology Standards Committee of the IEEE. “Draft Standard for Learning Object Metadata” IEEE 1484.12.1-2002. 15 July 2002

[2] Juan Diego Gutiérrez Gallardo. “XML – Manual imprescindible”. Anaya Multimedia. ISBN 84-415-1576-X

[3] Sitio Web de DCMI. http://dublincore.org/index.shtml. Consultado el 27 de febrero de 2009.

[4] Sitio web de la Biblioteca del Congreso Americano – Library of Congress. http://www.loc.gov/catdir/cpso/lcco/. Última vísita: 24 de junio de 2009

[5] Juan Ignacio Rouyet, Victor Martín. “A comparative study of the metadata in SCORM and Dublin Core”. Universidad Pontificia de Salamanca

[6] Joan Queralt Gil. “Tutorial para crear paquetes SCORM y usarlos en Moodle”. Enero 2005.

[7] Miguel Benito.”Sistemas de clasificación. Manual de aprendizaje de la Clasificación Decimal Universal y breve introducción a la Clasificación Decimal de Dewey”. Editorial Taranco, 1999.

Criptografía: El poder de lo oculto (y II)

Germán Carro Fernández
Rama de Estudiantes de la IEEE en la UNED.
Ingeniería Técnica en Informática de Sistemas.
Universidad Nacional de Educación a Distancia
A Coruña, España
E-mail: germancf@ieee.org

Abstract— **En este artículo se continúa con el análisis de la ciencia criptográfica, iniciado en el anterior (Págs. 49-61. Boletín nº 10, Rama de Estudiantes IEEE-UNED. 1 de Septiembre de 2008) mediante la descripción de los algoritmos MD5 y SHA, así como su filosofía, funcionamiento y utilidades. Finalmente se propone una solución implementada y diseñada; en lenguaje de programación JAVA; por el autor, con licencia de software libre, que permite describir y apreciar su funcionamiento práctico.**

Keywords- Criptografía; Software libre; MD5; SHA; PGP

I. INTRODUCCIÓN

Continuando en la línea abierta en nuestro artículo anterior, a lo largo de los párrafos que siguen, profundizaremos un poco más en lo que la criptografía supone en nuestros días.

Criptografía y códigos Hash, PGP, y su influencia actual son los temas que revisaremos en este recorrido por el mundo de los criptosistemas. Si bien nuestra primera aproximación se realizó desde un marco predominantemente histórico; necesario para comprender la evolución sufrida por estos sistemas en las últimas décadas; en este nuevo artículo intentaremos acercarnos a los mecanismos más utilizados hoy en día, para finalmente ilustrar con un ejemplo la facilidad con la que se puede implementar una aplicación software que plasme en la práctica el funcionamiento de los sistemas criptográficos.

II. CÓDIGOS HASH

Un código Hash nace de la aplicación de una función Hash a una cadena de caracteres. El funcionamiento es muy sencillo, la citada función generará una clave que representará unívocamente al documento a cifrar. Este algoritmo asegura que dos claves diferentes pertenecen a dos documentos diferentes, no obstante no nos garantizan el recíproco, de hecho uno de los problemas de las claves Hash radica en que dos claves iguales pueden pertenecer a documentos diferentes. La razón de esta debilidad se sustenta en que las claves están limitadas por cadenas de caracteres finitas en número inferior a la extensión y combinación de cadenas de caracteres de un documento a cifrar.

Aún así la probabilidad de colisiones[1] posibles es lo suficientemente aceptable para permitir que las funciones Hash se hayan convertido en las más habituales a la hora de pensar en el diseño de herramientas criptográficas. De hecho dos de las herramientas que utilizan estas funciones son MD5 y SHA-1:

- **MD5:** Es la abreviatura de "Algoritmo de Resumen del Mensaje 5". En esencia es un algoritmo de encriptación de 128 bits que utiliza funciones Hash. Fue diseñado por Ronald Rivest[2], del MIT[3], en 1991. Su mayor debilidad es que si bien en ciertas aplicaciones especializadas con un relativamente pequeño número de entradas que son conocidas de antemano es posible construir una función de Hash perfecta, que asegura que todas las entradas tengan una salida diferente, en una función en la cual se puede introducir datos de longitud arbitraria y que devuelve un Hash de tamaño fijo, como MD5, siempre habrá colisiones, ya que un Hash dado puede pertenecer a un infinito número de entradas. Para apreciar el funcionamiento del MD5 se adjunta a continuación, en pseudocódigo, un ejemplo del mismo en la Figura 1:

```
/* Procesar cada bloque de 16 palabras. */
para i = 0 hasta N/16-1 hacer

  /* Copiar el bloque 'i' en X. */
  para j = 0 hasta 15 hacer
    hacer X[j] de M[i*16+j].
  fin para /* del bucle 'j' */

 /* Guardar A como AA, B como BB, C como CC,
y D como DD. */
 AA = A
 BB = B
 CC = C
 DD = D

 /* Ronda 1. */
 /* [abcd k s i] denotarán la operación
     a = b + ((a + F(b, c, d) + X[k] + T[i]) <<< s). */
 /* Hacer las siguientes 16 operaciones. */
 [ABCD  0  7  1] [DABC  1 12  2] [CDAB  2 17  3]
[BCDA  3 22  4]
```

```
 [ABCD  4  7  5] [DABC  5 12  6] [CDAB  6 17  7]
[BCDA  7 22  8]
 [ABCD  8  7  9] [DABC  9 12 10] [CDAB 10 17
11] [BCDA 11 22 12]
 [ABCD 12  7 13] [DABC 13 12 14] [CDAB 14 17
15] [BCDA 15 22 16]

 /* Ronda 2. */
 /* [abcd k s i] denotarán la operación
    a = b + ((a + G(b, c, d) + X[k] + T[i]) <<< s). */
 /* Hacer las siguientes 16 operaciones. */
 [ABCD  1  5 17] [DABC  6  9 18] [CDAB 11 14
19] [BCDA  0 20 20]
 [ABCD  5  5 21] [DABC 10  9 22] [CDAB 15 14
23] [BCDA  4 20 24]
 [ABCD  9  5 25] [DABC 14  9 26] [CDAB  3 14
27] [BCDA  8 20 28]
 [ABCD 13  5 29] [DABC  2  9 30] [CDAB  7 14
31] [BCDA 12 20 32]

 /* Ronda 3. */
 /* [abcd k s t] denotarán la operación
    a = b + ((a + H(b, c, d) + X[k] + T[i]) <<< s). */
 /* Hacer las siguientes 16 operaciones. */
 [ABCD  5  4 33] [DABC  8 11 34] [CDAB 11 16
35] [BCDA 14 23 36]
 [ABCD  1  4 37] [DABC  4 11 38] [CDAB  7 16
39] [BCDA 10 23 40]
 [ABCD 13  4 41] [DABC  0 11 42] [CDAB  3 16
43] [BCDA  6 23 44]
 [ABCD  9  4 45] [DABC 12 11 46] [CDAB 15 16
47] [BCDA  2 23 48]

 /* Ronda 4. */
 /* [abcd k s t] denotarán la operación
    a = b + ((a + I(b, c, d) + X[k] + T[i]) <<< s). */
 /* Hacer las siguientes 16 operaciones. */
 [ABCD  0  6 49] [DABC  7 10 50] [CDAB 14 15
51] [BCDA  5 21 52]
 [ABCD 12  6 53] [DABC  3 10 54] [CDAB 10 15
55] [BCDA  1 21 56]
 [ABCD  8  6 57] [DABC 15 10 58] [CDAB  6 15
59] [BCDA 13 21 60]
 [ABCD  4  6 61] [DABC 11 10 62] [CDAB  2 15
63] [BCDA  9 21 64]

 /* Ahora realizar las siguientes sumas. (Este es el
incremento de cada uno de los cuatro registros por
el valor que tenían antes de que este bloque fuera
inicializado.) */
```

```
   A = A + AA
   B = B + BB
   C = C + CC
   D = D + DD

 fin para /* del bucle en 'i' */
          if ((cont%26)==0){
           rotar(rotor2,1);
          }
   if ((cont%676)==0){
    rotar(rotor3,1);
   }
   //Fin rotacion
   }
   // Forma 12 bloques de 5 char por línea
   if ((cont%5)==0)
    putc(' ',sa);
   if ((cont%60)==0)
    putc('\n',sa);
  }
 fclose(en);
 fclose(sa);

 printf("\n");
}
return 0;
}
```

Figura 1**.** Pseudocódigo de ejemplo del MD5 [5].

Una aplicación habitual de este tipo de algoritmo es la comprobación de que un archivo descargado de Internet no se ha alterado. Comparando un valor MD5 publicado con el que genera el archivo descargado, se garantiza que éste no ha sido modificado, protegiendo así al usuario de la existencia de virus informáticos en el mismo y garantizando que la descarga no se ha corrompido o ha quedado incompleta durante el proceso.

- **SHA:** Es la abreviatura de "Algoritmo de Hash Seguro". Constituye la denominación dada a la familia de algoritmos de encriptación que utilizan funciones Hash diseñadas por la Agencia Nacional de Seguridad de los Estados Unidos. Concretamente el SHA-1 ha sido examinado muy de cerca por la comunidad criptográfica pública y en el año 2004, un equipo de investigadores chinos, compuesto por Xiaoyun Wang, Yiqun, Lisa Yin y Hongbo Yu (principalmente de la *Shandong University* en China), demostró que eran capaces de romper el SHA-1 en al menos 2^{69} operaciones, unas 2000 veces más rápido que un ataque de fuerza bruta (que requeriría 2^{80} operaciones). Los últimos ataques contra SHA-1 han logrado debilitarlo hasta 2^{63}. El siguiente ejemplo en

pseudocódigo muestra el funcionamiento de SHA-1 en la Figura 2:

```
/*Nota 1: Todas las variables son de 32 bits y en módulo 2^32 cuando son calculadas.
Note 2: Todas las constantes en este pseudocódigo están en Big endian.
En cada palabra, el bit mas significativo es almacenado en la posición más a al izquierda.*/

/*Inicialización de variables:*/
h0 = 0x67452301
h1 = 0xEFCDAB89
h2 = 0x98BADCFE
h3 = 0x10325476
h4 = 0xC3D2E1F0

/*Preprocesado:*/
añadir el bit '1' al mensaje
añadir k bits '0', donde k es el mínimo número ≥ 0 tal que la longitud del mensaje resultado (en bits) sea congruente con 448(mod 512)
añadir longitud del mensaje (antes de preprocesarlo), en bits, como entero big-endian 64-bit

/*Procesar el mensaje en segmentos sucesivos de 512-bit:*/
romper mensaje en segmentos de 512-bit
for cada segmento
   romper segmento en dieciséis palabras big-endian de 32-bit w[i], 0 ≤ i ≤ 15

  /* Extender las dieciséis palabras de 32-bit a ochenta palabras de 32-bit:*/
   for i from 16 to 79
      w[i] = (w[i-3] xor w[i-8] xor w[i-14] xor w[i-16]) leftrotate 1

   /*Inicializar valor hash para este segmento:*/
   a = h0
   b = h1
   c = h2
   d = h3
   e = h4

   /*Bucle principal:*/
   for i from 0 to 79
      if 0 ≤ i ≤ 19 then
         f = (b and c) or ((not b) and d)
         k = 0x5A827999
      else if 20 ≤ i ≤ 39
         f = b xor c xor d
         k = 0x6ED9EBA1
      else if 40 ≤ i ≤ 59
         f = (b and c) or (b and d) or (c and d)
         k = 0x8F1BBCDC
      else if 60 ≤ i ≤ 79
         f = b xor c xor d
         k = 0xCA62C1D6

      temp = (a leftrotate 5) + f + e + k + w[i]
      e = d
      d = c
      c = b leftrotate 30
      b = a
      a = temp

   /*Añadir el hash de este segmento al resultado:*/
   h0 = h0 + a
   h1 = h1 + b
   h2 = h2 + c
   h3 = h3 + d
   h4 = h4 + e

/*Producir el valor final del hash (big-endian):*/
digest = hash = h0 append h1 append h2 append h3 append h4
```

Figura 2. Pseudocódigo de ejemplo del SHA-1.

Aplicaciones habituales de este tipo de algoritmos son, actualmente, las de firma digital que ya estuvimos analizando genéricamente en el primer artículo de esta serie sobre criptografía.

Vemos una vez más que la paciencia es el mejor aliado para descifrar estos sistemas criptográficos. En cualquier caso, MD5 y SHA son solo un par de ejemplos que demuestran la importancia de las funciones Hash en la evolución de la criptografía.

III. PGP

No es posible hablar de la criptografía moderna sin comentar algo acerca de PGP. Las siglas son la abreviatura de "Pretty Good Privacy". Este programa nació como un proyecto de software libre en el año 1991 de la mano del norteamericano Philip R. Zimmermann, y su primera fase de desarrollo activo llegó hasta diciembre de 2002, año en que el proyecto se escindió abiertamente en una vertiente comercial y dirigida a empresas y otra, alternativa, con soporte en software libre destinada a particulares. Ambas vertientes conviven de manera amigable y complementaria hoy en día y están supervisadas de una u otra forma por el propio Zimmermann.

El proyecto PGP se basa en el uso de criptografía de clave pública o criptografía asimétrica. El funcionamiento de este sistema se explica a continuación con un ejemplo:

Supongamos que el sujeto A quiere enviar un mensaje al sujeto B pero A no quiere que ningún sujeto C sea capaz de leer ese mensaje. En un entorno de clave pública el sujeto A posee la clave pública del sujeto B (CPubB), y el propio sujeto B posee además una clave privada que solo él conoce (CPrivB). Si el sujeto A encripta el mensaje con la clave pública de B (CPubB), éste podrá descifrar el mensaje de A al recibirlo, utilizando la clave privada (CPrivB) que sólo él posee. De la misma forma, en este contexto, si A y B se intercambian sus claves públicas (CPubB, CPubA) y utilizan sus claves privadas para encriptar sus mensajes respectivamente (CPrivB, CPrivA) siempre se podrá identificar al remitente como A o B ya que únicamente cada uno de ellos tendrá su propia clave privada desconocida para el otro pero vinculada a su respectiva clave pública.

La garantía de este proceso es lo que ha permitido que en la actualidad se utilicen los sistemas asimétricos para los procedimientos de firma electrónica. El problema de estos sistemas es el excesivo tiempo requerido para el proceso de encriptación y el incremento de tamaño en el documento cifrado motivado por el uso del mismo. Por este motivo habitualmente se utiliza una combinación de sistemas asimétricos y simétricos que garantice un equilibrio en tiempo, tamaño y seguridad que flexibilice el proceso.

PGP emplea de hecho este tipo de combinación y su éxito ha demostrado que puede utilizarse con seguridad en redes de información financiera, seguridad en redes de datos, protocolos de manejo de claves, sistemas operativos y redes de área local.

La seguridad de los protocolos y algoritmos utilizados por PGP es tan firme y dependiente de la generación de las claves asimétricas generadas que su código fuente es de acceso público y puede descargarse en varias páginas web relacionadas con el producto (ver sección de Referencias al final de este artículo).

De hecho, este sistema es tan efectivo que su uso en el código del "Conficker" ha convertido a éste en uno de los gusanos más difíciles de controlar en la historia de la seguridad informática. Como se puede ver, la criptografía y su eficiencia tienen aplicaciones y utilidades positivas y negativas como se ha demostrado a lo largo de toda su historia.

IV. UN EJEMPLO PRÁCTICO

Después de leer las líneas anteriores y el artículo precedente, se puede llegar a pensar que la criptografía es en sí mismo un galimatías difícil de entender y aún más de implementar de manera práctica. Nada más lejos de la realidad. De hecho es muy fácil establecer un algoritmo que siente las bases de un sistema de encriptación de datos y desarrollar un software con él. Para demostrarlo contamos con "Contego".

Contego es un software implementado en lenguaje JAVA y con licencia GPL-GNU. Existe versión para Linux (probado en distribuciones Debian y Ubuntu, empaquetado en un archivo *.*tar que sólo necesita ser descomprimido para poder utilizarlo) y para Windows (con un archivo autoinstalable *.*exe de fácil eliminación si se desea).

Este programa es muy simple en su funcionamiento, pero no por ello deja de ser una poderosa herramienta de encriptación. En este caso la versión Beta 1.1 solo permite realizar encriptación de datos en texto plano con formato Unicode UTF-8. Si bien el procedimiento está debidamente explicado en su sección de ayuda, podemos resumirlo diciendo que posee tres zonas claramente diferenciadas, que se pueden apreciar en la Figura 3 y que analizaremos a continuación:

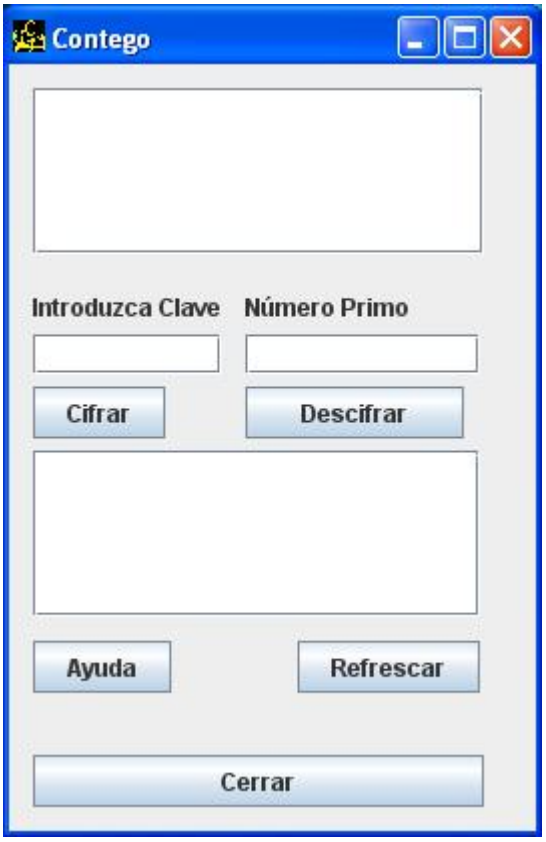


Figura 3. Imagen de Contego.

- La primera zona está constituida por un espacio para introducir el texto que posteriormente querremos encriptar.
- Una segunda zona contiene las secciones en las que introduciremos la clave de encriptación que queremos utilizar y un número primo que actuará como diferenciador de nuestro código encriptado. Una vez decididas ambas variables sólo tendremos que proceder a pulsar el botón "Cifrar".
- En tercer lugar tenemos un nuevo espacio de texto en el que aparece nuestro texto inicial una vez codificado y aplicado el cifrado de la sección anterior. De la misma forma, si queremos descifrar un texto, tendremos que escribirlo en esta tercera zona, introducir las variables "Clave" y "Número Primo" adecuadas y pulsar el botón "Descifrar",

transformando así el texto codificado en el mensaje original.

Un ejemplo del funcionamiento del programa se puede ver en la Figura 4.

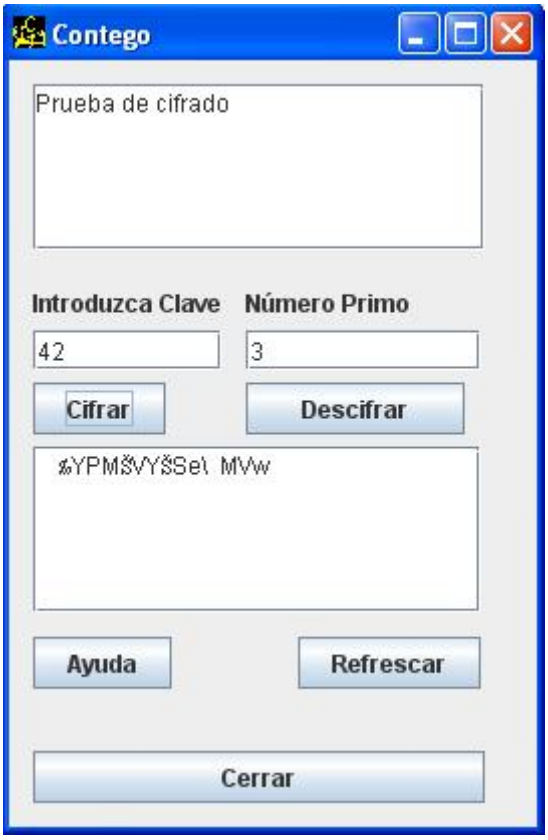


Figura 4. Ejemplo de cifrado.

Una vez realizado el cifrado solo debemos de copiar el texto que aparece en la tercera zona; teniendo buen cuidado de incluir los espacios y todos los caracteres que aparezcan en ella; y pegarlo en nuestro cliente de correo, o bien guardarlo en un archivo de texto plano para enviarlo como adjunto a través del correo electrónico. Debemos tener siempre la precaución, tanto al pegar el texto como al grabarlo en un archivo, de que siempre se esté utilizando el formato Unicode UTF-8, ya que si no, la desencriptación en destino producirá un resultado incorrecto. Evidentemente el destinatario deberá tener la clave y el número primo utilizado para proceder al encriptado de origen, y por supuesto una copia de Contego para desencriptar el mensaje.

Un ejemplo de descifrado se puede ver en las Figuras 5 y 6. Utilizando la "Clave" y "Número Primo" usados para el encriptado en origen, se procede a descifrar el código obteniéndose el mensaje que tendrá un texto idéntico al inicial.

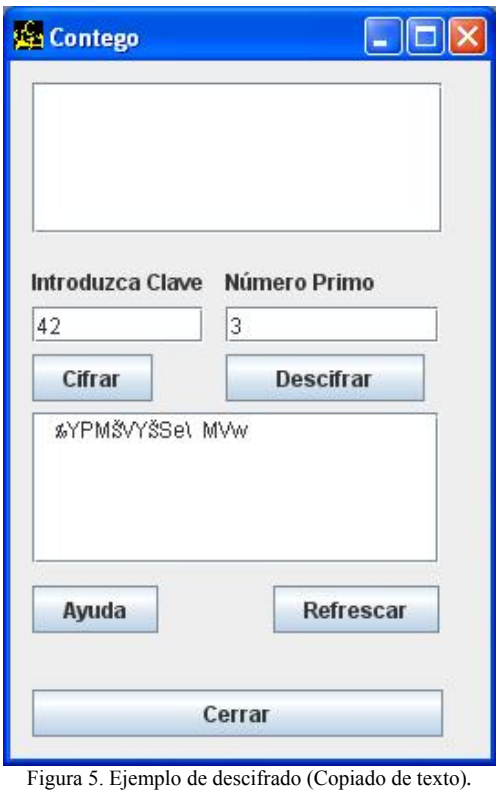


Figura 5. Ejemplo de descifrado (Copiado de texto).

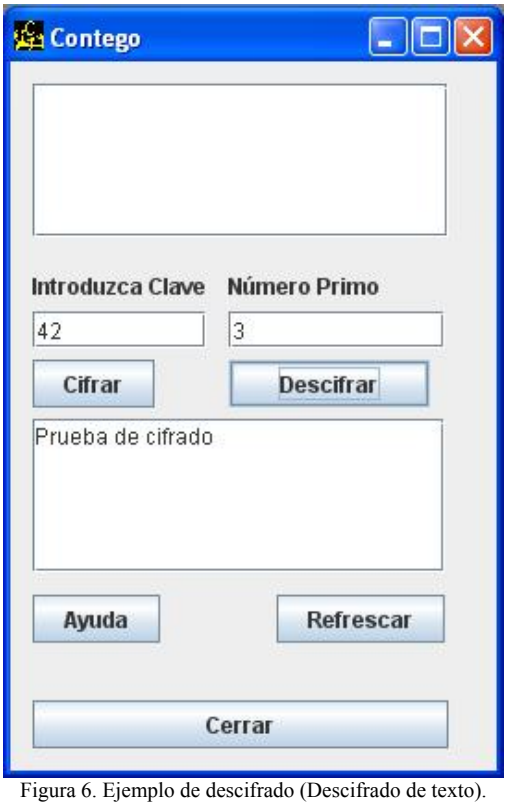


Figura 6. Ejemplo de descifrado (Descifrado de texto).

El funcionamiento externo es bastante explícito, pero ¿cómo trabaja realmente este software? ¿Qué algoritmo utiliza para realizar los procesos de encriptación y desencriptación de

texto? Pues bien, realmente no se ha buscado un algoritmo complejo y oscuro, se ha utilizado uno de los algoritmos más comunes y antiguos de la codificación para encriptación de claves: El cifrado por transformación afín. Este tipo de cifrado, también denominado cifrado monoalfabético es muy simple, y consiste en realizar la operación de sustitución de los caracteres del alfabeto. Puede obtenerse de forma matemática aplicando la siguiente expresión de equivalencia:

$$Ci = (a*Mi + b) \bmod n$$

en donde "a" se conoce como la constante de decimación, "b" es la constante de desplazamiento y n es el orden del grupo. Evidentemente "a" y "b" no son unos valores cualesquiera. En nuestro ejemplo "a" viene dado por el valor del número primo que decidamos utilizar para combinar con la clave. Por otro lado "b" es el desplazamiento dentro del alfabeto que iniciará el procesamiento de la clave imitando el comportamiento del cifrado César4. El código de ejemplo se muestra a continuación en la Figura 7:

```
...
private void
CifraButtonActionPerformed(java.awt.event.ActionEve
nt evt) {
       String Clave=(ClaveTextField.getText());
       String Primo=(PrimoTextField.getText());
       String Origen=(TextoOrigenTextArea.getText());
       int NuevoTexto=0;
       String DeNuevo="";

       if (Clave.matches("[0-9]*")&Primo.matches("[0-
9]*")){b = (int) ((Double.parseDouble(Clave)));
       // "a" es la constante de decimación.
       a = (int) ((Double.parseDouble(Primo)));

       // Cifrado por transformación afín
       // Ci=(a*Mi+b)mod n;
       // Mi=(a^-1)(Ci+n-b)mod n;

       for (int x=0; x<Origen.length(); x++)
       {
         //Este 256 permitirá una clave de hasta 256.
          NuevoTexto = (a*((int)Origen.charAt(x))+
b)%256;

         DeNuevo=(DeNuevo+(char)NuevoTexto);
       }
        if (Clave.matches("[0-9]*")) {int clavenum = (int)
((Double.parseDouble(Clave)));
       TextoDestinoTextArea.setText
(String.valueOf(DeNuevo));
       }}
        else
       {

JOptionPane.showMessageDialog(ErrorDialog,"Introdu
zca
número","Error",JOptionPane.ERROR_MESSAGE);
       }

private void
DescifraButtonActionPerformed(java.awt.event.Action
Event evt) {
       String Clave=(ClaveTextField.getText());
       String Primo=(PrimoTextField.getText());
       String Destino=(TextoDestinoTextArea.getText());
       int NuevoTexto2=0; String DeNuevo2="";
       if (Clave.matches("[0-9]*")&Primo.matches("[0-
9]*")){b = (int) ((Double.parseDouble(Clave)));

       // "a" es la constante de decimación.
       a = (int) ((Double.parseDouble(Primo)));

       while((((i*256)-1)%a)!=0) {i++;}
       n=(((i*256)-1)/a);
       m= (256-n);
       for (int x=0; x<Destino.length(); x++)
       {
         NuevoTexto2
=(m*(((int)Destino.charAt(x))+256-b))%256;
         DeNuevo2=
DeNuevo2+(char)NuevoTexto2;//Llave.charAt(NuevoT
exto2-23);
       }
       if (Clave.matches("[0-9]*")) {int clavenum = (int)
((Double.parseDouble(Clave)));
       TextoDestinoTextArea.setText
(String.valueOf(DeNuevo2));
       }}
      else{

JOptionPane.showMessageDialog(ErrorDialog,"Introdu
zca
número","Error",JOptionPane.ERROR_MESSAGE);
       }
    }
…
```

Figura 7. Código de ejemplo de encriptación y desencriptación por cifrado afín en JAVA. Extraído del código fuente de Contego.

La función CifraButtonActionPerformed se encarga de realizar el pertinente cifrado de nuestro texto y por otro lado la función DescifraButtonActionPerformed permitirá descifrar el código anterior con éxito. El procedimiento de creación y análisis criptográfico debe siempre estar vinculado a la encriptación y desencriptación. Siempre que se proceda a diseñar un algoritmo para encriptar un texto o documento con

éxito no debemos olvidar que luego querremos recuperar esa información y ésta debe ser idéntica a la información original. Si no es así nuestro algoritmo no será viable y la recuperación de la información inicial será fragmentada, incompleta o distinta a la de origen, con lo que no nos será útil y corromperá el principio de seguridad y fiabilidad que rige la criptografía.

Por otro lado alguien puede preguntarse ¿si ya sabemos cuál es el algoritmo utilizado para encriptar un texto, qué seguridad tiene ahora este programa?. En realidad la seguridad sigue siendo la misma que antes de conocerse el código fuente, de hecho la fuerza de un algoritmo criptográfico se demuestra precisamente dándolo a conocer para que cualquiera pueda dirigir "ataques" contra él para reducir el tiempo de descifrado a través de diferentes combinaciones utilizando otros algoritmos para ello. En nuestro caso, a pesar de usar un algoritmo muy sencillo, la capacidad criptográfica se mantiene, ya que tanto la Clave como el Número Primo a utilizar son lo suficientemente aleatorios; al ser decididos por el usuario; y permiten realizar combinaciones que complican la desencriptación por fuerza bruta5 lo suficiente como para que alguien tenga que estar muy seguro de la importancia de la información a descifrar o tenga mucho tiempo libre para dedicarse a ello.

Es evidente que en este proceso entra en juego la psicología (utilizar una y otra vez la misma clave, recurrir a fechas familiares, etc.) que puede acabar con cualquier sistema de seguridad por muy fiable que éste sea, siempre que se utilice un programa o algoritmo dependiente del usuario para diseñarlo. De ahí la importancia; cuando hablamos de seguridad; de no confiar únicamente ésta a sistemas informáticos, con independencia de la garantía que éstos nos ofrezcan, y pensar que en el proceso de custodia de datos, nosotros y nuestro comportamiento, somos el primer muro de protección.

V. CONCLUSIONES FINALES

Llegamos al final de este artículo. Es fundamental entender que el área de la criptografía no puede explicarse sólo con los apuntes que hemos desmenuzado en éste y en el artículo anterior, es necesario profundizar en todos los conceptos analizados y pensar que la perspectiva que debemos de tener nos obliga a revisar la historia y el uso de los diferentes instrumentos; hardware y software; que han servido a lo largo de ésta para ir desarrollándola.

Se ha intentado, eso sí, que aquellas personas interesadas vean cubierta su curiosidad y dejen de percibir esta ciencia como algo oscuro y siniestro. No es difícil adentrarse en el mundo de la seguridad, pero si existen conceptos y conocimientos que deben de estar claros antes de iniciarse en ella porque si no es así corremos el riesgo de caer en un laberinto de algoritmos que pueden desanimarnos antes de empezar a conocer realmente este maravilloso mundo de la encriptación y desencriptación de datos.

Ya en el primer artículo vimos la importancia actual que la seguridad tiene en un mundo virtualizado cada vez más, como es el nuestro, y la que ha tenido históricamente. En este segundo artículo hemos analizado un poco más a fondo alguno de los algoritmos más utilizados y, sobre todo, se ha querido demostrar la facilidad con la que cualquier persona con conocimientos básicos de programación puede diseñar su propio programa de encriptación y desencriptación de texto plano para utilizarlo en sus documentos y comunicaciones diarias. Y, una vez, más se ha insistido en la importancia y responsabilidad que el propio individuo tiene a la hora de garantizar la seguridad en sus propias comunicaciones.

VI. NOTAS ACLARATORIAS

1.- ***Colisión de Hash***: es una situación que se produce cuando dos entradas distintas en una función de Hash producen la misma salida. Es matemáticamente imposible que una función de Hash carezca de colisiones, ya que el número potencial de posibles entradas es mayor que el número de salidas que puede producir un Hash. Sin embargo, las colisiones se producen más frecuentemente en los malos algoritmos.

2.- ***Ronald Rivest*** *(1947-)*: Criptógrafo y; actualmente; profesor de ciencias de la computación en el departamento de ingeniería eléctrica y ciencias de la computación del MIT. Es muy conocido por su trabajo con el cifrado de clave pública junto con Len Adleman y Adi Shamir, específicamente en el algoritmo RSA, con el que ganaron en el año 2002 el premio ACM Turing.

3.- ***MIT***: El **Instituto Tecnológico de Massachusetts** (**MIT**, del inglés *Massachusetts Institute of Technology*) es una de las principales instituciones dedicadas a la docencia y a la investigación en Estados Unidos, especialmente en ciencia, ingeniería y economía. El Instituto está situado en Cambridge, Massachusetts, y cuenta con numerosos premios Nobel entre sus profesores y antiguos alumnos. MIT es considerado como uno de los mejores centros universitarios de ciencia e ingeniería del mundo

4.- ***Cifrado César***: En criptografía, un cifrado César, es una de las técnicas de codificación más simples y más usadas. Es un tipo de cifrado por sustitución en el que una letra en el texto original es reemplazada por otra letra que se encuentra tres posiciones más adelante en el alfabeto. Por ejemplo, la A sería sustituida por la D (situada 3 lugares a la derecha de la A), la B sería reemplazada por la E, etc. Este método debe su nombre a Julio César, que lo usaba para comunicarse con sus generales (ver también ***"Criptografía: El poder de lo oculto (I)."*** Págs. 49-61. Boletín nº 10, Rama de Estudiantes IEEE-UNED. 1 de Septiembre de 2008).

5.- ***Desencriptación por fuerza bruta***: En criptografía, se denomina ataque de fuerza bruta a la forma de recuperar una clave probando todas las combinaciones posibles hasta encontrar aquella que permite el acceso. Dicho de otro modo, define al procedimiento por el cual a partir del conocimiento del algoritmo de cifrado empleado y de un par texto claro/texto cifrado, se realiza el cifrado (respectivamente, descifrado) de uno de los miembros del par con cada una de las posibles combinaciones de clave, hasta obtener el otro miembro del par.

SOBRE EL SOFTWARE

El software utilizado como ejemplo en estas páginas ha sido diseñado, implementado y creado por Germán Carro

Fernández; autor de este artículo. Su nombre es "Contego" y está amparado por la licencia GPL-GNU. Cualquier persona que lo desee puede usarlo y modificarlo, respetando los términos de esa licencia, descargando gratuitamente una copia; tanto para Windows como para Linux; de los enlaces en la página oficial del programa:

http://contegosoftware.sourceforge.net/

O bien del entorno aLF:

Espacio en aLF de la Rama de Estudiantes del IEEE en la UNED (http://www.innova.uned.es)

También existe un repositorio en Savannah (perteneciente al "*Project GNU*"de software libre) del que se puede descargar una copia para Linux en:

http://savannah.gnu.org/submissions_uploads/Contego.tar

Y, por supuesto, se puede solicitar una copia del mismo enviando un e-mail a: contego.cripto@gmail.com indicando en el Asunto: "Copia Windows" o "Copia Linux", respectivamente (o ambos en su caso). El software se encuentra en su fase Beta con lo que se ruega encarecidamente leer la "Ayuda" y la sección "Acerca de Contego" antes de usarlo.

REFERENCIAS

[1] J. Ramió Aguirre. "Aplicaciones Criptográficas". Departamento de Publicaciones de la Escuela Universitaria de Informática de la Universidad Politécnica de Madrid. España. 1999.

[2] E. Garcia, M. A. Lopez, J. J. Ortega. "Una introducción a la Criptografía". Departamento de Matemáticas E.S. de Informática de la Universidad de Castilla-La Mancha. 2005.

[3] G. Lawton, "On the Trail of the Conficker Worm." Computer, Jun. 2009, pp. 19-22. IEEE Computer Society Publications.

[4] http://www.kriptopolis.org

[5] http://en.wikipedia.org/wiki/SHA_hash_functions

[6] http://www.wikipedia.org/

[7] http://people.csail.mit.edu/rivest/

[8] http://telematica.cicese.mx/gaceta/pgp.html

[9] http://www.pgpi.org/

[10] http://www.philzimmermann.com

Inteligencia Artificial en la Composición Musical Asistida por Ordenador (CAO)

Emerson Castañeda Sanabria
Dpto. Ingeniería del Software e Inteligencia Artificial.
Fac. de Informática, Universidad Complutense de Madrid
Madrid, España
emecas@ieee.org

Abstract—**Este artículo presenta parte de los conceptos recopilados tras el estudio y la evaluación de diferentes técnicas relacionas con la inteligencia artificial actualmente empleadas para la composición musical asistida por ordenador (CAO). Del estudio de un total de 12 técnicas se seleccionaron algunas de las más relevantes para incluirlas en el presente trabajo. Para cada una de las técnicas se incluye una definición, los tópicos relacionados, referencias de los trabajos más representativos y el modelo propuesto. La representación del modelo esta orientada a esclarecer los potenciales aportes a la propuesta de una arquitectura integradora para la CAO.**

Keywords- IA, Vida Artificial, Computación Evolutiva, Algoritmos Genéticos, Autómatas Celulares.

I. INTRODUCCIÓN

La variedad de trabajos que relacionan la inteligencia artificial con la música es bastante extensa. Dentro de estos podemos encontrar un subconjunto en los que se aborda de una forma parcial o total la composición musical asistida por ordenador. Tras la realización de un estado del arte actualizado sobre este subconjunto de estudios se realiza un minucioso estudio de las técnicas asociadas con la inteligencia artificial que tienen relación con la composición musical.

Aquí se presentan los conceptos para la evaluación de las diferentes técnicas que son actualmente empleadas o están en miras su aplicación en la composición musical asistida por ordenador (CAO). Del estudio de un total de 12 se seleccionaron algunas de las más relevantes para incluirlas en el presente trabajo. Para cada una de las técnicas se incluye una definición, algunos de los tópicos relacionados, referencias de los trabajos más representativos y el modelo inicial que representa sus potenciales aportes orientados a la propuesta de una arquitectura para la CAO que las integre.

II. VIDA ARTIFICIAL (ARTIFICIAL LIFE - ALIFE)

A. *Definición*

La vida artificial es el estudio de la vida y de los sistemas artificiales que exhiben propiedades similares a los seres vivos, a través de modelos de simulación. C. Langton fue el primero en utilizar el término cuando durante la "1ra Conferencia Internacional de Síntesis y Simulación de Sistemas Vivientes" (también conocida como Vida Artificial I) en Los Alamos National Laboratory en 1987.

B. *Referencias sobre Vida Artificial*

En [12] se presenta la vida artificial como una disciplina que estudia los sistemas vivientes naturales para similar algunos de sus aspectos biológicos en el silicio[1]. Miranda afirma que en el intento de imitar fenómenos biológicos sobre ordenadores proporciona una ruta viable para un mejor entendimiento de la teoría de los organismos vivos, mas aun en las aplicaciones practicas de los principios biológicos para la tecnología y la medicina. Dado que la vida artificial trata con estos fenómenos complejos, su desarrollo ha fomentado un conjunto de herramientas de investigación para el estudio de la complejidad; por ejemplo, los autómatas celulares, los algoritmos genéticos, los juegos adaptativos y las redes neuronales. Algunas de estas herramientas están también probadas en otros campos diferentes al de la biología: en el campo de las ciencias sociales[2] la lingüística[3] y la musicología[4]

Otra aplicación destacable no precisamente en el campo de la música, pero que si despierta un interés particular es el programa Infoartrópodos[5] desarrollado por Ernesto J. Carmena, este utiliza círculos y rayas de distintos colores para representar las variaciones genéticas, Los genes son unos valores que representan las dimensiones de las formas que irá

- [1] cit. C. G. Langton. Artificial Life: an overview. Cambridge (USA): The MIT Press. 1997.
- [2] cit. G. N. Gilbert y K. G. Troitzsch. Simulations for the Social Scientist. Buckingham (UK): Open University Press. 1999.
- [3] cit. L. Steels. Syntetic Modeling od Language Origins. Evolutions of Communications Journal 1 Nro 1, pp 1-34, 1997.
- [4] cit. P. Todd. Simulating the Evolution of Musical Behaviour. En The Origins of Music, N. l. Walling, B. Merkr, y S. Brown (eds), Cambridge (USA): The MIT Press. 2000.

  cit. E. Bilota et al. Synthetic Harmonies an approach to musical semiosis by jeans of cellular automata. Artificial Live VII: proceedings of seventh International Conference on Artificial Live, M. A. Bedu et al (ads). The MIT Press. 2000.
- [5] Una versión para DOS disponible en: http://www.fortunecity.com/underworld/fifa/613/vida.zip y una versión en línea: http://come.to/webensis [Consulta: 21/09/2009].

tomando el Infoartrópodos. El usuario va seleccionando el Infoartrópodos que generará la siguiente generación.

La Sociedad Internacional para la Vida Artificial (ISAL - *The International Society for Artificial Life*)[6] fue fundada en mayo de 2001 como una institución no lucrativa. ISAL es una sociedad democrática, internacional, profesional dedicada a promover la educación e investigación científica referente a la vida artificial, incluyendo el patrocinio de congresos, publicación de revistas científicas (*journals*) y boletines de prensa, y el mantenimiento de sitios Web relacionados con la vida artificial. *Artificial Life* (publicada por MIT Press) es la publicación oficial de ISAL, y *The International Conference on Artificial Life* es la reunión oficial científica de la sociedad.

En el libro [12] hay un capitulo dedicado a lo que el autor llama Música Evolucionista o Evolutiva (*Evolutionary Music*), en el que se tratan algunos de los aspectos relacionados con al vida artificial y la música. El autor afirma que la vida artificial puede tener múltiples aplicaciones para la investigación en musicología, donde pero quizás el enfoque hacia la composición sea la aplicación más interesante, prestándose para el estudio de las circunstancias y los mecanismos por medio de los cuales la música puede originarse y desenvolverse en mundos creados artificialmente habitados por comunidades virtuales de músicos y oyentes[7]. Como paradigmas de la vida artificial Miranda presenta: los autómatas celulares, los algoritmos genéticos, y los juegos adaptativos, el primero se tratan con más detalle en la sección correspondiente a la técnica de autómatas celulares y los dos restantes en la sección sobre computación evolutiva.

Como una rama asociada a la vida artificial encontramos la robótica. En [14] se discute el desarrollo de músicos automatizados que interpretan sobre instrumentos reales a través del uso de dispositivos mecánicos como servomotores y solenoides. El trabajo trata sobre la construcción del hardware y la programación del software, así como también algunas innovaciones de investigación asociando música, robótica y ciencia de la computación[8].

En [13] se revisa la teoría y las aplicaciones de los algoritmos de hormigas, y de nuevos métodos de optimización discreta basados en la simulación de una colonia auto organizada de hormigas biológicas. La colonia puede ser considerada como un sistema multiagentes donde cada agente funciona independientemente por reglas simples. A diferencia del comportamiento casi primitivo de los agentes, el comportamiento del sistema entero acierta en ser asombrosamente razonable. Los algoritmos de hormigas han sido extensamente estudiados por investigadores europeos desde mediados de los años noventa. Estos algoritmos han sido aplicados exitosamente para solucionar muchos problemas combinatorios complejos de optimización, tales como el problema del viajante de comercio, el problema de determinación del recorrido del vehículo, el problema cuadrático de asignación, el problema de optimización de tráfico de redes, etc. Los algoritmos de hormigas son especialmente eficientes para la optimización en línea de procesos en sistemas no estacionarios distribuidos (por ejemplo, la determinación de los recorridos en la red de telecomunicación).

En [7] se describen cómo es posible generar música simulando movimientos de hormigas artificiales en una gráfica donde los vértices representan las notas y los bordes representan transiciones posibles entre notas. Como las hormigas pueden depositar feromonas en las orillas, ellas colectivamente construyen una melodía que es una secuencia de eventos MIDI. Aquí son probados diferentes ajustes de parámetros para producir estilos diferentes de música generada con varios instrumentos. También se introduce un mecanismo para inicializar la matriz de feromona desde archivos musicales.

C. Modelo propuesto para la Vida Artificial

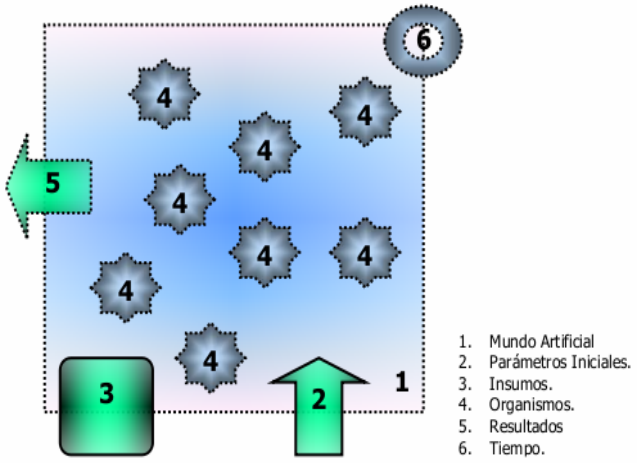


Figura 1. Modelo propuesto para la técnica Vida Artificial

III. COMPUTACIÓN EVOLUTIVA (EVOLUTIONARY COMPUTATION)

La computación evolutiva nace en el año de 1993 y retoma conceptos de la evolución y la genética para resolver principalmente problemas de optimización. Esta rama de la inteligencia artificial tiene sus raíces en tres desarrollos relacionados pero independientes entre si: Algoritmos genéticos, Programación evolutiva, y Estrategias Evolutivas.

Los algoritmos genéticos fueron desarrollas por John H. Holland en la década de 1960 y su motivación inicial fue la de proponer un modelo general de proceso adaptable. La programación evolutiva fue creada en la década de 1960 y su creador fue L. J. Fogel. Este desarrollo comenzó como un esfuerzo encaminado a crear inteligencia artificial basado en la evolución de máquinas de estado finitas. Las estrategias

- [6] Disponible en linea en: http://www.alife.org
- [7] Estudiando los orígenes y la evolución en el contexto de las convenciones culturales que pueden emerger bajo un número de restricciones, por ejemplo sicológicas, fisiológicas o ecológicas.
- [8] Para una mejor apreciación de este trabajo, varias piezas musicales interpretadas por el músico automatizado así como también algunos clips de vídeo están disponibles para descarga en http://www.bridgeport.edu/sed/projects/IFACWeb/default.html

evolutivas fueron propuestas por Ingo Rechenberg y Hans-Paul Schwefel en la década de 1970. Su principal objetivo era el de resolver problemas de optimización de parámetros.

La computación evolutiva toma como base las ideas de la evolución propuestas por C. Darwin y en los descubrimientos realizados por G. Mendel en el campo de la genética.

A. Algoritmos Genéticos

Los algoritmos genéticos comprenden métodos de computación inspirados por procesos biológicos, más exactamente aquellos procesos que son considerados a ser las fuerzas conductoras del origen y evolución de las especies, los cuales propuso C. Darwin sobre 1850 [3] y más recientemente Richard Dawkins [4]. Como en el caso de los autómatas celulares y las redes neuronales, los algoritmos genéticos también representan ciertas metáforas y jerga de la biología.

Los algoritmos genéticos son normalmente empleados para encontrar soluciones óptimas en ingeniería y problemas de diseño donde puede existir una solución alternativa, pero su evaluación no puede ser determinada hasta que esta ha sido implementada y probada. En tales casos, la solución óptima puede ser buscada por una generación manual. Implementando y probando las alternativas o por aproximaciones hechas de manera gradual mejorando las soluciones no óptimas encontradas. De esta manera la complejidad del problema en cuestión aumenta, sin embargo, estos procedimientos llegan a ser cada vez más insatisfactorios y malgastadores. Por lo tanto la necesidad de un método automático que explore la capacidad del los ordenadores para realizar voluminosas tareas combinacionales. No obstante, los algoritmos genéticos van más allá de un proceso combinacional estándar siendo un poderoso mecanismo para apuntar a combinaciones fructíferas potenciales. Estos mecanismos se asemejan a aquellos de la evolución biológica tales como la selección natural basada en aptitud (*fitness*), cruce de genes, mutación y demás; por lo tanto el titulo de algoritmos genéticos.

En la secuencia típica de un algoritmo genético, se crea una población inicial de entidades abstractas de forma aleatoria. Dependiendo de la aplicación estas entidades pueden representar prácticamente cualquier cosa, desde los componentes fundamentales de un organismo hasta los comandos de un robot o las notas musicales de una secuencia. Luego, se aplica un proceso de evaluación en función tanto de probar como de encontrar los objetivos para solucionar la tarea o problema en cuestión. Como esta población inicial puede errar en la evaluación, el sistema se embarca en la creación de una nueva generación de entidades. Primeramente, el número de entidades se fija aparte de los criterios preestablecidos. Estos criterios a menudo se refieren a su adaptabilidad para la reproducción puesto que este subconjunto experimentara un proceso de cruce con el objetivo de producir descendencia. Los criterios de aptitud obviamente varían su forma entre aplicación y aplicación pero en general indican que entidades de la actual generación trabajan mejor. La entidades seleccionadas son entonces combinadas (usualmente en pares) y dan nacimiento a la descendencia. Durante este proceso de reproducción, la formación de la descendencia involucra un proceso de mutación. A continuación, la descendencia es incluida en la población. El destino de las entidades restantes de la población no seleccionadas para la reproducción puede diversificar. Pero estas usualmente mueren saliendo sin causar ningún efecto. Este es el punto donde se dice que una nueva generación de la población ha evolucionado. El proceso de evaluación se aplica ahora a la siguiente generación. Si la población aun no logra los objetivos, entonces el sistema se embarca en la creación de una nueva generación. Este ciclo se repite hasta que la población pase la prueba de evaluación.

En la práctica, los algoritmos genéticos normalmente operan sobre un conjunto de códigos binarios que representan las entidades de la población. Estos involucran operaciones de tres clases básicas: recombinación, mutación y selección. Mientras el proceso de recombinación causa el intercambio de información entre un par de códigos, el proceso de mutación altera el valor de un simple bit en un código. La recombinación produce códigos descendientes por combinación de la información contenida en los códigos de sus padres. Dependiendo de la forma de representación de los códigos, pueden aplicarse dos tipos de recombinación: recombinación de valores reales o cruce de valores binarios.

B. Métodos de codificación

En orden de hacer un uso efectivo de los algoritmos genéticos se han desarrollado métodos tanto para codificar la población como para asociar el comportamiento del proceso evolutivo al dominio de aplicación, en nuestro caso: la música. La regla general dice que debe emplearse un alfabeto de código lo más pequeño posible para representar la población.

Un método típico de codificación es el de la codificación por cadenas binarias, por medio del cual cada individuo se representa por una cadena de una longitud específica. Este método de codificación es interesante dado que cada dígito del código, o grupos de dígitos, pueden asociarse con diferentes atributos del individuo.

El número de variaciones de la cadena codificada puede proyectarse, por ejemplo, se puede determinar códigos usando cadenas binarias largas divididas en palabras de longitud fija; donde cada palabra puede corresponderse con un valor decimal. En este caso, el código es una cadena decimal pero el cómputo del proceso del algoritmo genético ocurre sobre su correspondiente representación binaria.

C. Mecanismos de selección

El mecanismo de selección de un subconjunto de la población para su reproducción varía de acuerdo a la aplicación del algoritmo genético. Este mecanismo generalmente involucra la combinación de un método de asignación de de aptitud (*fitnees*) y un esquema de probabilidad. Uno de los mecanismos de selección más simples es el de selección estocástica de muestras (*stochastic sampling selection*).

Otro mecanismo de selección ampliamente usado es la selección local de vecindario (*local neighbourhood selection).* En este caso, los individuos son considerados para interactuar solo con la gama de un vecindario limitado. El vecindario por consiguiente define grupos de potenciales padres.

Con el propósito de hacer el rendimiento del proceso más efectivo, primero el algoritmo selecciona un grupo de candidatos adecuados (aleatoriamente por ejemplo) y entonces define un vecindario local para cada candidato. El colega para aparearse es seleccionado de este vecindario de acuerdo a su criterio *fitness*.

Los esquemas de vecindario generalmente usados son vecindarios: anillo, bidimensional y tridimensional, pero se pueden emplear esquemas más complejos. La distancia entre los vecinos no necesariamente tiene que ser unitaria y no todos los vecinos inmediatos deben considerarse.

Existen muchos más mecanismos de selección pero no es posible examinar todos aquí. Las aplicaciones exitosas de los algoritmos genéticos a menudo usan mecanismos especiales a la medida de la tarea.

D. Referencias sobre computacion evolutiva

En [10] se presenta una visión pragmática con miras a encontrar procedimientos singulares que puedan producir subjetivamente resultados interesantes en el empleo de métodos evolucionistas para la generación de música y arte. En este trabajo se definen cinco problemas abiertos y clarificados como amplias áreas de principios de investigación para el arte y la música evolucionista. Cada problema es explicado (la importancia y los antecedentes) y descrito en el contexto de sistemas evolucionistas creativos. Antes de introducirse en los problemas planteados el autor hace distinción sobre dos enfoques de investigación. Investigación donde la música resultante y el trabajo artístico pretender ser reconocido por los humanos como algo creativo (o sea el arte) e investigación que explora el concepto de creatividad en general.

El primer caso los resultados generados por el sistema y/o la metodología están dirigidos al ser apreciados por el humano como arte. Es decir, exhiben propiedades que la humanidad reconoce como despliegue de alguna forma de intención creativa o juicio estético por parte del creador (ya sea por persona o máquina). El autor dice que personas que usan tales técnicas pueden considerarse además de investigadores, artistas. El segundo caso es diferente. Aquí, la creatividad se considera en un contexto más manifiesto, y no se encuentra limitada a ser reconocida por las personas como algo creativo o estético. Es decir, la creatividad no se encuentra exclusivamente en el comportamiento humano.

Luego de clarificar lo anterior McCormack hace una exhaustiva revisión de los problemas abiertos los cuales se mencionan a continuación:

Problema abierto #1: Idear un sistema donde el genotipo, y el fenotipo y el mecanismo que produce fenotipo desde el genotipo sea capaz de modificarse de forma automática y robusta, hacer selección, y por lo tanto evolucionar.

Problema abierto #2: Idear funciones formalizadas de aptitud que son capaces de medir propiedades estéticas humanas de fenotipos. Estas funciones deben ser máquina representables y prácticamente computable.

Problema abierto #3: Crear sistemas EMA que produzcan arte reconoció por las humanos como contribución art*ística (a* distinción de cualquier fetiche puramente especializado o fascinación).

Problema abierto #4: Crear ecosistemas artificiales donde los agentes creen y reconozcan su propia creatividad. La meta de este problema es ayudar a entender la creatividad y el surgimiento, investigar las posibilidades del arte tal como puede ser.

Problema abierto #5: Para desarrollar teorías *de arte* de arte evolucionista y generativo.

E. Referencias sobre Algoritmos Genéticos

Como se menciono en el principio de esta sección, dentro del escenario de la computación evolutiva existe una herramienta ampliamente difundida, se trata de los algoritmos genéticos. Aquí se traen a colación algunos trabajos sobre composición musical en los han sido empleados los algoritmos genéticos con diferentes enfoques. Entre estos se pueden mencionar [11][8][15] donde se han usado en la harmonización de melodías, [9] donde son empleados para la composición autónoma, en el desarrollo de un sistema interactivo para la interpretación de improvisaciones de jazz en [1], para la extracción de patrones en piezas musicales monofónicas en [6], y en [5] donde son usados para la composición de fugas mediante la generación automática de contrapunto.

En [12] se ejemplifica el trabajo del compositor Gary Lee Nelson[9] en el que se usan los algoritmos genéticos para evolucionar patrones rítmicos. En este caso, se emplea una cadena binaria para representar una serie espacial de pulsos iguales que se articulada si el bit esta encendido. La prueba de aptitud está basada sobre una prueba de adición; si el número de bits que están encendidos es mayor a cierto umbral, entonces la cadena cumple la prueba de aptitud. Para valores altos de umbral se obtienen ritmos con mucha densidad. Inversamente, un valor bajo umbral tiende a producir texturas delgadas dirigidas a un completo silencio. Miranda también describe Vox Populli[10], un programa para la composición con algoritmos genéticos, donde los algoritmos genéticos son empleados para evolucionar conjuntos de acordes de acuerdo a un criterio especifico de *fitness* especificado por el usuario en términos de melodía, armonía y características del rango de voz.

Otro de los trabajos a destacar, que también se detalla en la sección de referencias correspondiente a la técnica de autómatas celulares, es [2]. Una de las principales aportaciones de este trabajo, es la forma de representar el material musical que se produce a partir de la interpretación de la configuración del autómata celular como una serie (*cromosoma* como lo llaman los autores) para realizar una búsqueda mediante un algoritmo genético que encuentran las mejores composiciones

- [9] cit. Gary Lee Nelson. Sonomorphs: An application of genetic algorithms to the growth and development of musical organisms. In Proceedings of the Fourth Biennial Art and Technology Symposium, pp 155-169. 1995.
- [10] Programa desarrollado por Jonatas Manzolli y Artemis Moroni. Ganador en The Dream Centenary Computer Graphics Prix 99 en Auzi, Japon.

musicales. El procedimiento seguido es el siguiente: A. Se inicia a partir de material del autómata celular representado en forma de serie, B. se generan varios individuos (secuencias de sonidos) asociados con estos cromosomas, C. se seleccionan las secuencias del cromosoma más adecuadas para evaluar las capacidades de los individuos (para este caso la secuencia de sonidos más consonante) D. se hacen evolucionar estos cromosomas pasando de una generación a otra, casualmente modificando sus características y usando combinación de reglas basadas en la geminación sexual, E. se continua el proceso por muchas generaciones.

F. Modelo propuesto para la Computacion Evolutiva

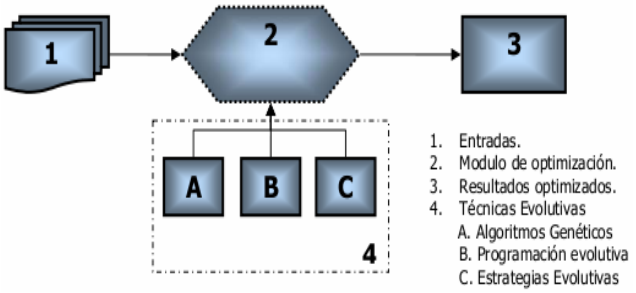


Figura 2. Modelo propuesto para la técnica Vida Artificial

IV. AUTÓMATAS CELULARES (CELLULAR AUTOMATON)

A. Definición

Un autómata celular es un modelo discreto estudiado en la teoría de la computación, las matemáticas y la biología para modelar sistemas que cambian algunas de sus características con el tiempo en donde el espacio y el tiempo son discretos, y toman cantidades sobre un conjunto finito de valores discretos. Consiste en una retícula de celdas finita, donde cada una de las celdas puede tener un número finito de estados. La retícula puede tener cualquier dimensión finita. El tiempo se maneja de forma discreta y el estado de una celda en el tiempo *t* es función del estado de un número finito de sus celdas vecinas en el tiempo *t-1*. Todas las celdas tienen las mismas reglas para su actualización, basándose en los valores de sus vecinas. las reglas se evaluan para producir una nueva generación.

El hecho que una celda del CA pueda representar cualquier cosa desde una simple variable numérica hasta unidades sofisticadas de procesamiento hace al CA una poderosa herramienta de modelado. Aquí hacemos solo referencia al tipo de aplicación de los CA en donde las celdas mantienen variables simples que pueden contener un número entero. Esta clase de autómata celular estudiado es a menudo referenciada como un autómata celular *p-state* debido a que sus celdas pueden tomar valores p de los valores 0, 1, 2, 3, …, p-1.

B. Ejemplos de Autómatas Celulares

Al día de hoy hay muchos ejemplos de algoritmos para CA disponibles en [12] se presentan dos ejemplos[11] llamados El juego de la vida (Game of life) inventado por John Horton Conway y Demon Cyclic Space diseñado por David Griffeath. Los algoritmos se detallan a continuación.

C. El juego de la vida

El juego de la vida (Game of Life) es un CA de dos dimensiones que intenta modelar una colonia de organismos virtuales simples. En teoría, el autómata se definiría como una cuadricula infinita, pero por propósitos prácticos normalmente se define como un arreglo de celdas de n x m, donde cada una de las celdas tiene dos posibles estados: *viva* representado por el numero 1 o *muerta* representado por el numero 0, coloreadas de negro o blanco respectivamente.

El estado de las celdas en el tiempo t es determinado por su estado en el tiempo t-1 y el estado de sus celdas vecinas también en el tiempo t-1. Hay 4 reglas esenciales para determinar el destino de las celdas en el siguiente tiempo t.

3) Nacimiento (Birth): una celda que está muerta en el tiempo t vivirá en el tiempo t+1 si exactamente tres de sus celdas vecinas están vivas en el tiempo t.

4) Muerte por sobrepoblación (Death by overcrowding): una celda que está viva en el tiempo t estará muerta en el tiempo t+1 si cuatro a más de cuatro de sus vecinas están vivas en el tiempo t.

5) Muerte por exposición (Death by exposure): una celda que está viva en el tiempo t estará muerta en el tiempo t+1 si tiene una o ninguna celda vecina viva en el tiempo t.

6) Supervivencia (Survival): la celda que está viva en el tiempo t permanecerá viva hasta el tiempo t+1 solo si dos o tres de sus vecinas están vivas en el tiempo t.

D. Demon Cyclic Space

Este es otro ejemplo de autómata celular en donde las celdas asumen más de dos estados. Cada uno de los cuales está representado por un color diferente y se numera de 0 hasta n-1. Las reglas para que opere la evolución son las siguientes: una celda en cierto estado k en un tiempo t es dominada por el estado de las celdas adyacentes que están en estado k-1, significa que las celdas adyacentes cambian desde el estado k al estado k-1. Esta regla asemeja una cadena natural en donde la celda en estado 2 puede dominar a la celda en estado 1 inclusive si la última domina una celda en estado 0. Esta cadena no termina debido a que la automatización es cíclica, lo que significa que una celda en estado 0 domina sus vecinas en estado n-1. Este CA genera mundos en miniatura complejos. Aun iniciando desde una distribución de celdas coloreadas aleatoriamente, siempre terminara en estabilidad.

- [11] cit. A. K. Dewdney. A cellular universe of debris, droplets, defects and demons. Scientific American, 261, Num 2, pp 102-105. 1989.

E. Referencias sobre Autómatas Celulares

Un trabajo muy representativo dentro de los que emplean los CA para la composición musical es CAMUS[12]. Un programa para generar música usando autómatas celulares, fue desarrollado originalmente por Eduardo Miranda para plataforma ATARI y rediseñado e implementado para plataforma Windows por Kenny McAlpine en dos versiones CAMUS y CAMUS 3D.

La primera versión del CAMUS par la plataforma ATARI cuenta con un tutorial en línea[13] dispuesto por el autor y donde también se puede descargar una versión gratuita del mismo. CAMUS emplea un modelo cartesiano para representar una tripleta; es decir, un trío de notas. El modelo tiene dos dimensiones, donde la coordenada horizontal representa el primer intervalo de la tripleta y la coordenada vertical representa su segundo intervalo, esto se puede apreciar mejor en la figura 3. Cada tiempo t del juego de la vida produce un número de tripletas

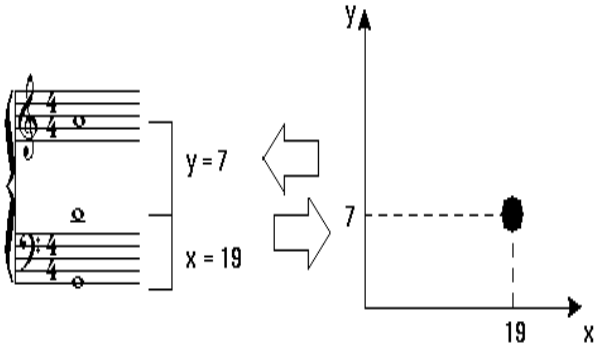


Figura 3. Representación de un trío de notas en CAMUS [12]

El sistema usa los dos autómatas en paralelo para producir música. El autómata del Juego de la vida produce tripletas y el autómata Demon Cyclic Space determina la orquestación de la composición. En este caso, cada color corresponde a un instrumento (MIDI) designado para interpretar las notas generadas por una celda específica. Cada celda musical tiene su propio cronometraje, pero las notas dentro de una celda pueden asumir duraciones diferentes y pueden ser disparadas en momentos diferentes.

Para iniciar el procesamiento de la música, el autómata del juego de la vida se establece con una configuración inicial, y el autómata Demon Cyclic Space se inicializa con condiciones aleatorias, luego ambos se ponen a correr. En cada paso de tiempo, las coordenadas de cada celda viva del primer autómata son analizadas y usadas para determinar una tripleta que se interpretara en el tiempo correspondiente de la composición. Es estado de la celda correspondiente en el autómata Demon Cyclic Space se usa para determinar la orquestación de la pieza. Esta configuración está en la figura 4.

En este caso, la celda del juego de la vida en la posición (5, 5) está viva, y genera un evento acústico (que es, un conjunto de tres notas). La celda correspondiente en el Demon Cyclic Space está en condición 4, lo que quiere decir que el evento acústico será interpretado por el cuatro del instrumento (Usando el canal MIDI 4). Las coordenadas (5, 5) describen los intervalos en una tripleta: El tono fundamental, la nota cinco semitonos por encima del tono fundamental, y la nota diez semitonos por encima de tono fundamental.

Una vez que se ha determinado la tripleta para cada celda, las condiciones de las celdas vecinas en el juego de la vida se usan para calcular la duración y posición temporal de cada nota, según un conjunto de códigos temporales. Estos códigos determinan la forma temporal de cada tripleta; Los valores actuales para la activación y los parámetros de duración se calculan según una ecuación creada por el usuario.

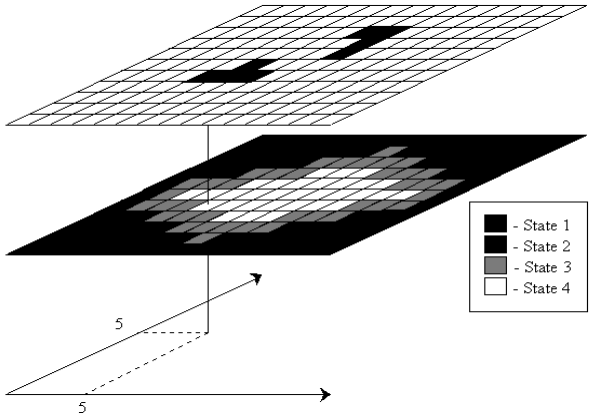


Figura 4. Una configuración de los autómatas de CAMUS [12]

Un segundo trabajo es [2] donde los autores exploran la creación de universos artificiales que son expresables mediante la música como sistemas complejos a través de sistemas de codificación y empleando lenguaje musical, es de esta manera como se logra entender los patrones que la dinámica global producida por los autómatas celulares y usar los resultados en el dominio musical. En este acercamiento los autores afirman que la música puede considerarse la semántica de la complejidad. A su vez identifican analogías entre elementos de los autómatas celulares y los elementos de forma musical, creando a un framework musical narrativo mediante el cual desarrollan lo que ellos llaman una *metodología computacional semántica*. Argumentándolo en que la música fomenta la capacidad para analizar y reconstruir complejidad, proveyendo una compenetración inesperada en su organización.

También se plantean las siguientes características para la creación de música a partir de vida artificial: (i) reglas variables para su producción, (ii) un número infinito de posibles producciones, (iii) un carácter común y abstracto de producción, (iv) una representación y codificación arbitraria, (v) múltiples semánticas de representación, (vi) posibilidades de múltiples lecturas de sus espacios matemáticos de configuración, (vii) patrones múltiples, tanto locales como globales, con relaciones para otros patrones, y (viii) múltiples formas de comportamiento evolucionista. Características que

- [12] El nombre CAMUS proviene de **C**ellular **A**utomata **MUS**ic generator.
- [13] Disponible en http://tamw.atari-users.net/camus/camustut.htm [Consultado: 21/07/2009]

los autores intentar plasmar en su música empleando los CA para la producción de material musical junto con los Algoritmos Genéticos encargados aquí de detectar las mejores composiciones musicales por medio de un mecanismo automático de selección que usa un criterio de aptitud basado en la consonancia.

El anterior trabajo se ve materializado en el programa Harmony Seeker[14] descrito en [12] aquí se combinan los autómatas celulares y los algoritmos genéticos para generar música. Básicamente el programa usa los autómatas celulares para generar el material musical que es analizado cuidadosamente por el algoritmo genético con la función de optimizar la salida del sistema.

F. Modelo propuesto para los Autómatas Celulares

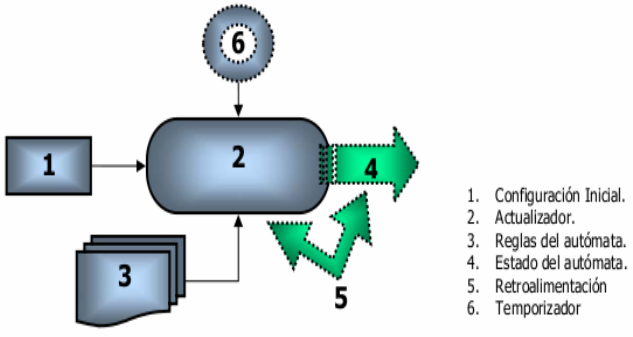


Figura 5. Modelo propuesto para los Autómatas Celulares

V. CONCLUSIONES Y TRABAJO FUTURO

El objetivo general consistió en estudiar y verificar hasta que punto eran aplicables a la disciplina de la Composición Musical, 12 técnicas relacionadas con la Inteligencia Artificial, ya ampliamente usadas en otras áreas para la construcción de software, proporcionar un conjunto de modelos orientados a la obtención de un prototipo de herramienta que facilitase las tareas del compositor musical empleando la fusión de las técnicas estudiadas, a partir de una arquitectura propuesta. Con el propósito de fusionar varias técnicas e intentar demostrar las posibilidades potenciales para la composición asistida por computador.

En este trabajo se presenta una parte de los conceptos recopilados tras el estudio y la evaluación de diferentes técnicas relacionas con la inteligencia artificial y actualmente empleadas para la composición musical asistida por ordenador. Se seleccionaron algunas de las más relevantes para incluirlas. Para cada una de las técnicas se incluyo una definición, algunos de los tópicos relacionados, referencias de los trabajos más representativos y el modelo inicial. La representación propuesta para el modelo inicial está orientada a esclarecer los potenciales aportes a la propuesta de una arquitectura integradora para la composición asistida por ordenador.

Las técnicas incorporadas en esta primera parte fueron Vida artificial (Artificial Life), Computación evolutiva (Evolutionary Computation), y Autómatas Celulares (Cellular automaton), el estudio estas técnicas y de las técnicas restantes permite establecer una clasificación donde se identifican las categorías de aplicaciones dentro de la CAO.

A partir de la anterior clasificación es posible identificar con más exactitud los tipos de datos involucrados dentro de cada una de las técnicas tras asociarles unas aplicaciones específicas. Luego de identificar con un nivel más minucioso los datos involucrados en el empleo de cada una de las técnicas para las diferentes categorías de aplicación propuestas, resulta posible proponer un modelo que permita tipificar sus potenciales de aplicación de tal manera que se pueda hacer referencia a las técnicas desde el punto de vista del proceso o los procesos para los que puede contribuir de una forma más eficiente, es decir los objetivos para los que una técnica puede resultar más productiva.

El trabajo futuro consiste en el diseño y la construcción de un del framework que esté acorde con una propuesta de arquitectura para el desarrollo de aplicaciones de CAO con técnicas de IA, que se pueda llevar a cabo de forma incremental. Implementando según las necesidades cada uno de los componentes requeridos para los distintos prototipos. La construcción de los componentes específicos, dada la cantidad de técnicas incorporadas en la arquitectura, ocasiona que la implementación del framework represente el llevar a cabo varios subproyectos con objetivos claramente condicionados de acuerdo a las distintas técnicas y orientados conseguir la "sinergia" entre las combinaciones de unas técnicas u otras.

- [14] Desarrollado por Valerio Talarico del Evolutionary Systems Group de la Universidad de Calabria, Italia.

REFERENCIAS

[1] J. Biles. GenJam: A genetic algorithm for generating jazz solos. ICMC'94 Proc. The International Computer Music Association. 1994.

[2] E. Bilotta, P. Pantano. Synthetic harmonies: an approach to musical semiosis by means of cellular automata. Leonardo Vol. 35, Nro. 2, pp 153-159, 2002.

[3] C. Darwin. On the origins of species by means of natural selection or the preservation of favoured races in the struggle for life. London Murray. 1858.

[4] R. Dawkins. The Blind Watchmaker. London, Penguin Books, 1986.

[5] A. Garay. Fugue Composition with Counterpoint Melody Generation Using Genetic Algorithms. Presentado en: Second Internacional Symposium Computer Music Modeling and Retrieval, Esbjerg, Dinamarca May 2004 en LNCS 3310, Uffe Kock (Ed) pp. 96-106, 2004.

[6] C. Grilo, A. Cardoso. Musical Pattern Extraction Using Genetic Algorithms. . Presentado en: Internacional Symposium Computer Music Modeling and Retrieval, Montpellier, France mayo 2003, en LNCS 2771, Uffe Kock (Ed) pp 114-123, 2003.

[7] C.Gueret, N. Monmarche, y M. Slimane. Ants Can Play Music. ANTS 2004, LNCS 3172, pp. 310–317, 2004.

[8] A. Horner, L. Ayers. Harmonisation of musical progression with genetic algorithms. ICMC'95 Proceedings, San Francisco: International Computer Music Association, pp 483-484. 1995

[9] A. Horner, D. E. Goldberg. Genetic algorithms and Computer Assisted Music Composition. ICMC'91 Proceedings, San Francisco: International Computer Music Association, pp 479-482, 1991.

[10] J. McCormack. Open Problems in Evolutionary Music and Art. EvoWorkshops 2005, LNCS 3449, pp. 428–436, 2005.

[11] R. A. McIntyre. Composition with genetic algorithms. Technical report, University of Michigan. 1995.

[12] E.R. Miranda. Composing Music with Computers. Focal Press – Music Technology Series. 2001.

[13] S. D. Shtovba. Ant Algorithms: Theory and Applications. Programming and Computer Software, Vol. 31, No. 4, 2005, pp. 167–178. Translated from Programmirovanie, Vol. 31, No. 4, 2005.

[14] T. M. Sobh, B. Wang, y K.W. Coble. Experimental Robot Musicians. Journal of Intelligent and Robotic Systems Volume 38 , Issue 2 , pp 197-212. 2003

[15] G. Wiggins, G. Papadopoulos, S. Phon-Amnuaisuk, A. Tuson. Evolutionary methods for musical composition, 1998. disponible en: http://www.dai.ed.ac.uk/daidb/papers/documents/rp882.html. [Consulta: 21/09/2009].

English Zone

Gloria Murillo
Coordinadora Boletín Electrónico e English Zone
Universidad Nacional de Educación a Distancia
Madrid, España

I. UNTIL VS UP TO?

Empleamos "until" para decir hasta con una referencia en el tiempo "hasta" (día, mes, hora o acción en concreto).

Tienes hasta el próximo viernes para terminar el proyecto.	You've got **until** next Friday to finish the project.
No me iré hasta que llegues.	I won't go **until** you arrive.
Se quedará allí hasta que se muera.	He will stay there **until** he dies.
No lo sabré hasta que no se publiquen los resultados.	I won't know **until** the results are published.
Te esperaremos hasta las cinco.	We'll wait for you **until** five o'clock.

Sin embargo, cuando queremos decir **"hasta"** referido a **un punto físico** (por ejemplo una cumbre) o a un **periodo de tiempo** se suele decir **"up to"**.

A veces tengo que esperar hasta 30 minutos.	Sometimes I have to wait for **up to** 30 minutes.
Llegó hasta la cumbre sin oxígeno.	He climbed **up to** the summit without oxygen.
Tuvimos que reenviar el documento hasta 20 veces.	We had to resend the document **up to** 20 times.
Tienes seis meses para mejorar tu inglés.	You have **up to** six months to improve your English.
El agua me llegó a las rodillas.	The water came **up to** my knees.

II. ALLUDE OR ELUDE?

Allude means to mention something or someone indirectly.

Elude means to escape from something or someone, to evade.

- **allude**
  The candidate alluded to the recent scandal during the interview. (=He talked about it indirectly.)
  He didn't want to say who was fighting, but he alluded to our children.
  Some parts of the New Testament allude to parts of the Old Testament.

- **elude**
  The candidate eluded the recent scandal during the interview. (=He didn't talk about it.)
  The actress eluded her fans by going out through the back door.
  They eluded the teacher and ran away to the street.

III. ONE ADJECTIVE

Self-confident- seguro de si mismo

Why are you so self-confident?

¿Por qué estas tan seguro de ti mismo?

IV. ONE EXPRESSION

Rome wasn´t built in a day.

No se gano Zamora en una hora.

V. ONE PHRASAL VERB

To take down – apuntar

Take down my name and number

ISSN: 1989-2195

Apunta mi número de telefono.

VI. WHAT´S IT LIKE?

"What's it like?" es una pregunta muy común que puede prestar alguna confusión debido a la palabra "like". En realidad no tiene nada que ver con el verbo **"like"** (el verbo aquí es "to be"). Se usa para decir **"¿qué tal?"** o **"¿cómo es?"** cuando preguntamos sobre algo.

¿Cómo está el tráfico en el centro de la ciudad?	**What's** the traffic **like** in the city centre?
¿Qué tiempo hace donde estás?	**What's** the weather **like** where you are?
¿Cúal es la situación en Oriente medio en este momento?	**What's** the situation in the Middle East **like** at the moment?
¿Qué tal tu libro?	**What's** your book **like?**
¿Cuáles son las perspectivas en tu trabajo nuevo?	**What are** the prospects **like** in your new job?

Nota: Cuando contestamos a este tipo de preguntas, nunca empleamos “Like”. Ver los siguientes ejemplos en pasado.

VII. FUNNY LESSON.

Un chiste en lengua inglesa que nos hará aprender cierto vocabulario en inglés y nos hará esbozar alguna sonrisa

Adultery

There's this old priest who got sick of all the people in his parish who kept confessing to adultery. One Sunday, in the pulpit, he said, "if I hear one more person confess to adultery, I'll quit!"

Well, everyone liked him, so they came up with a code word. Someone who had committed adultery would say they had "fallen". This seemed to satisfy the old priest and things went well, until the priest died at a ripe old age.

About a week after the new priest arrived, he visited the Mayor of the town and seemed very concerned. The priest said, "you have to do something about the sidewalks in town. When people come into the confessional, they keep talking about having fallen."

The Mayor started to laugh, realizing that no-one had told the new priest about the code word. Before the mayor could explain, the priest shook an accusing finger at the mayor and said, "I don't know what you're laughing about, Your wife fell three times this week”.

Vocabulary

accusing finger *(akiúsin fínguer)* - dedo acusador
adultery *(adólteri)* - adulterio
confessional *(conféshonal)* - confesionario
parish *(párish)* - parroquia
pulpit *(pólpit)* - púlpito
ripe *(ráip)* - maduro
sidewalk *(sáid-uók)* - vereda, acera
someone *(sámoan)* - alguien
priest *(príist)* - cura, sacerdote

Phrasal verbs

to come up with - inventar
to get sick - enfermarse, hartarse

VIII. LINKS

[1] http://www.funnylessons.com/
[2] http://www.mansioningles.com/index.htm
[3] http://www.vausys.com/
[4] http://www.vaughanradio.com/reproductor/player2.htm
[5] http://www.saberingles.com.ar/songs/315.html

Rama de Estudiantes UNED
http://www.ieec.uned.es/ieee-uned/

Hazte socio
de la Rama de Estudiantes
del IEEE en la UNED

Web IEEE-UNED

http://www.ieec.uned.es/ieee-uned/
Más info: elio@ieec.uned.es

Charlas, conferencias,
cursos, visitas, empresa,
Boletín Electrónico, etc.