Volume 2, Issue 2
November 2015
ISSN: 1390-9266

LAJC

LATIN-AMERICAN JOURNAL OF COMPUTING

FACULTAD DE INGENIERÍA DE SISTEMAS

QUITO - ECUADOR

Editorial Committee:

PhD. Jenny Torres, National Polytechnic School, Ecuador
PhD. Edison Loza, University of Grenoble, France
PhD. Ricardo Villalón, University of Costa Rica, Costa Rica

http://lajc.epn.edu.ec/

ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

LATIN AMERICAN JOURNAL OF COMPUTING

LAJC

Vol II, Issue 2, November 2015
ISSN: 1390-9266

Published by:
National Polytechnic School
Faculty of Systems Engineering
Department of Informatics and Computer Sciences

Quito-Ecuador

LATIN AMERICAN JOURNAL OF COMPUTING - LAJC

Published by:

National Polytechnic School
Faculty of Systems Engineering
Department of Informatics and Computer Sciences
Ecuador

Editorial Committee:

PhD. Jenny Gabriela Torres Olmedo, National Polytechnic School, Ecuador
PhD. Edison Loza, University of Grenoble, France
PhD. Ricardo Villalón, University of Costa Rica, Costa Rica

Editor in chief:

PhD. Jenny Gabriela Torres Olmedo, National Polytechnic School, Ecuador

Section Editors:

Eng. Hernán David Ordoñez Calero, National Polytechnic School, Ecuador
Ms. Sandra Milena Nazamués Quenguán, National Polytechnic School, Ecuador
Mr. Santiago Alejandro Sandoval Hinojosa, National Polytechnic School, Ecuador

Mailing address:

Escuela Politécnica Nacional, Facultad de Ingeniería de Sistemas
Ladrón de Guevara E11-253, La Floresta
Quito - Ecuador, Apartado Postal: 17-01-2759

Web address:

http://lajc.epn.edu.ec/

E-mail:

lajc@epn.edu.ec

Frequency:

3 issues per year

Circulation:

500

EDITORIAL

Dear readers,

We are pleased to present the second issue of the Latin American Journal of Computing – LAJC in 2015. This issue includes five articles covering different aspects of Information Security, Information Systems, Intelligent Systems and Software Engineering.

Latin American Journal of Computing - LAJC is an open access peer-reviewed journal, sponsored by the Faculty of Systems Engineering of the National Polytechnic School of Ecuador, which goal is to bring together researchers and practitioners from academia and industry to present original research papers, technical reviews and state-of-the-art reviews for publication.

We would like to thank all authors, who contributed to the success of the Journal. Special thanks to the reviewers for their contributions to keeping the high quality of the selected papers.

Cordial thanks are due to the Section Editors members for their efforts and the organizational work. Finally, we cordially thank National Polytechnic School for supporting and publishing this volume.

We hope that you enjoy reading this issue and find the articles informative and useful. We highly encourage you to submit your work, which fit within the scope of LAJC. Please keep in mind that we have an open call for submissions twice a year.

For detailed instructions on the preparation and submissions of manuscripts, please check the URL below:

http://lajc.epn.edu.ec/index.php/LAJC/pages/view/call-for-papers

We will be happy to receive your comments and feedback on our journal.

Best Regards,

PhD Jenny Torres
Editor in chief
LAJC

Latin American Journal of Computing - LAJC
Reviewers

We are most grateful to the following individuals for their time and commitment to review manuscripts for Latin American Journal of Computing - LAJC for this edition.

Aguiar Pontes Josafá, PhD. National Polytechnic School, Ecuador
Anchundia Carlos, MSc. National Polytechnic School, Ecuador
Barriga Jhonattan, MSc. National Polytechnic School, Ecuador
Benalcázar Marco, PhD. National Polytechnic School, Ecuador
Brandão Diego, PhD. Fluminense Federal University, Brazil
Calle Tania, MSc. National Polytechnic School, Ecuador
Fuertes Díaz Walter, PhD. Army University - ESPE, Ecuador
Hallo María, MSc. National Polytechnic School, Ecuador
Intriago Monserrate, MSc. National Polytechnic School, Ecuador
Loza Aguirre Edison, PhD. University of Grenoble, France
Lucio José Francisco, PhD. National Polytechnic School, Ecuador
Magreñán Ángel Alberto, PhD. International University La Rioja, Spain
Navarrete Rosa, MSc. National Polytechnic School, Ecuador
Paz Arias Henry, MSc. National Polytechnic School, Ecuador
Pérez María, PhD. National Polytechnic School, Ecuador
Pousa Federico, PhD. University of Buenos Aires, Argentina
Riofrío Daniel, PhD. University of New Mexico, United States
Roa Henry, PhD. The University of Queensland, Australia
Sánchez Gordón Sandra, MSc. National Polytechnic School, Ecuador
Torres Olmedo Jenny, PhD. National Polytechnic School, Ecuador
Zambrano Patricio, MSc. National Polytechnic School, Ecuador

TABLE OF CONTENTS

Método para Adaptar una Librería de Processing a la Web
Method for Adapting a Processing Library to the Web

Cesar Colorado y Jean Pierre Charalambos

***Resumen*— Se presenta un método para la adaptación de libre- rías aportadas por usuarios del lenguaje de gráficos Processing, basado en Java, al lenguaje de gráficos para la web Processing.js, basado en HTML5, WebGL y JavaScript. Se revisan diversos métodos para hacer adaptaciones a la web. En nuestro enfoque, proponemos crear una arquitectura que permite que la librería aportada por el usuario, se compile de Java a Javascript, usando la tecnología Google Web Toolkit, evitando modificar la librería del usuario y haciendo la adaptación en un solo trunk de desarrollo. La arquitectura tiene tres capas: la librería del usuario, una capa que simula el comportamiento de Processing y una para utilizar la librería en la web. Se exponen dos prototipos de librerías adaptadas.**

***Palabras clave*— Google Web Toolkit, GWT, HTML5, JavaScript, Processing, Processing.js, WebGL, Web Graphics.**

***Abstract*— A method for the adaptation of libraries done by users for the Processing graphics language, based on Java, to the graphics engine for the web processing.js, based on WebGL and JavaScript is presented. Various methods to make adaptations to the web are reviewed. In our approach, we propose to create a architecture that is compiled to JavaScript, using Google Web Toolkit technology, in order to maintain the user library without modifications and making the adaptation in a single trunk of development. The architecture has three-tiers: the user library, a layer that simulates the Processing behavior and a layer to use the user library in the web. It is presented two prototypes of adapted libraries.**

***Index Terms*—Google Web Toolkit, GWT, HTML5, JavaScript, Processing, Processing.js, WebGL, Web Graphics.**

I. INTRODUCTION

PROCESSING es un lenguaje y entorno de programación de código abierto basado en Java , para la creación de imágenes, animaciones e interacciones; está orientado a diseñadores, artistas digitales, estudiantes e investigadores[1]. Buena parte de la funcionalidad del lenguaje es extendida mediante librerías contribuidas por miembros de la comunidad.

Los programas de Processing (denominados sketches) en su versión estable 1.5.*, se pueden ejecutar dentro de un navegador web mediante la tecnología de Applets de Java [2], lo que supone el inconveniente de tener que instalar en el navegador una extensión propietaria y no estándar. En el pasado un enfoque similar ha sido empleado por otras tecnologías como Adobe Flash [3], Microsoft Silverlight [4] y X3D [5].

Processing.js [6], es una adaptación de Processing en JavaScript (JS en adelante) para la web, basado en el elemento canvas de HTML5 y WebGL [7]. Este enfoque fue recientemente adoptado en la versión actual de desarrollo de Processing (2.2.1 en el momento de este escrito)[8], para la ejecución de un sketch cualquiera dentro del navegador, dejando en desuso la tecnología de Applets de Java. Processing.js soporta la mayoría de las funciones de Processing-1.5.*; pero su versión actual adolece (v-1.4.1 al momento de este escrito) de dos inconvenientes: 1. No soporta varias de las funciones introducidas en Processing-2.0b8, particularmente las concernientes al manejo de shaders; y 2. No cuenta con un mecanismo para incluir librerías de terceros (mecanismo que estaba presente en la tecnología de Applets de Java), limitando de manera considerable su uso.

Nuestro objetivo es atender el segundo de los problemas, proponiendo un método con el cual se puedan adaptar librerías de terceros a Processing.js, poniéndolas a disposición del programador final dentro de un contexto web, con una mínima intervención de su parte (ver Sección III y IV). Además, el método se puede adaptar fácilmente para desarrolladores Java con poco conocimiento en JS (ver Sección V). Para demostrar la validez de nuestro enfoque, dos librerías Processing escritas en Java, han sido portadas a JS (ver Sección VI). Finalmente, discutiremos las limitaciones y alcances futuros del método (ver Sección VII).

II. TRABAJOS PREVIOS

Revisando los sketches y librerías para Processing.js en [9], podemos distinguir dos tipos de métodos para realizar una adaptación de un sketch de Processing a la web: automático y

Cesar Colorado, Departamento de Ingeniería de Sistemas e Industrial, Facultad de Ingeniería, Universidad Nacional de Colombia, e-mail: cacolorador@unal.edu.co

Jean Pierre Charalambos, Departamento de Ingeniería de SIstemas e Industrial, Facultad de Ingeniería, Universidad Nacional de Colombia, e-mail: jpcharalambosh@unal.edu.co

manual. En el tipo método automático se utiliza un programa que traduzca el sketch de Java a JS, tal como lo hace el modo JS de Processing 2.0. El tipo de método manual consiste en tomar el algoritmo del sketch y escribirlo en JS.

A. *Adaptación manual*

Existen algunas librerías de Processing adaptadas a JS, por ejemplo Toxiclibs.js [8], [10], que es un ejemplo donde se reescribe clase por clase el código de Java a JS, conservando la estructura de los paquetes, las firmas de los métodos y los algoritmos originales [10].

B. *Processing 2.0 Modo Javascript*

En Processing 2.0 es posible ejecutar un sketch en un navegador web con el MODO JAVASCRIPT [8], que utiliza la función parseProcessing de Processing.js. El modo JS puede adaptar sketches simples; pero no puede realizar la adaptación de librerías más grandes o complejas, debido a que no es posible depurar ni soporta las utilidades y ventajas de Java, como POO, interfaces, colecciones, etcétera.

Sea usando el modo JS Mode o realizando la adaptación manual, es requerido abrir una rama nueva de desarrollo y la directa intervención del desarrollador, haciendo la adaptación altamente susceptible a errores.

III. ENFOQUE PROPUESTO

Como se mencionó, el objetivo de nuestro método es adaptar las librerías de Processing a la web modificando lo menos posible el código fuente original y sin requerir abrir una rama nueva de desarrollo. Proponemos una arquitectura de tres capas para mantener el código fuente original intacto, donde se traducen las librerías de Java a JS automáticamente utilizando el compilador Google Web Toolkit (GWT) [11]. La adaptación está limitada a usar las clases del lenguaje Java definidas en el GWT Java Runtime Environment Emulation Reference (JRE ER)[12], sin embargo podemos utilizar muchas de las utilidades del lenguaje como colecciones, programación orientada a objetos, enums, tipos genéricos, además del depurador.

A. *Alcance y objetivos de la adaptación*

El esquema general del método se ve en la figura 1 y cumple dos objetivos:

1. La librería portada a JS debe tener una API muy similar o idéntica a la de la librería en Java.
2. La librería portada a JS debe poder realizar los mismos llamados a la API de Processing que a la librería en Java.

B. *Arquitectura del método*

El código compilado a JS de GWT es ofuscado y usa una sintaxis propia [11]. Para que este pueda ser utilizado por el usuario, es necesario crear un "wrapper" de JS sobre la librería en Java. En nuestra implementación, se construyen tres capas jerárquicas donde irán el wrapper y el acceso a la API de Processing de esta manera:

- Core: El código original de la librería.
- Facade: El wrapper que pública a la capa Core en un objeto JS. Cumple el objetivo 1.
- Back: Empleando la terminología de Java RMI [13], esta capa tiene clases Stub que simulan el comportamiento de Processing y llaman en su lugar a Processing JS. Cumpliendo el objetivo 2.

Esta arquitectura se ve en la Figura2.

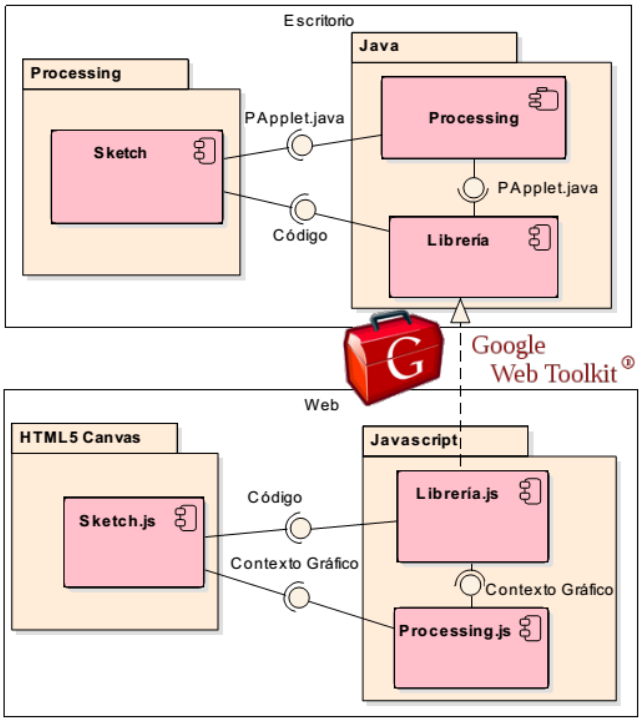


Figura 1. Adaptación de una librería de Processing

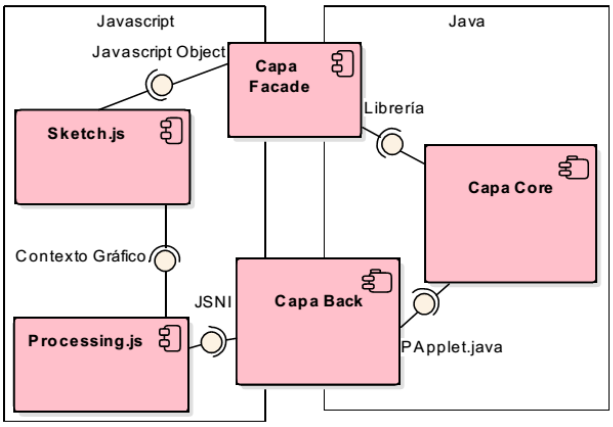


Figura 2. Arquitectura de tres capas

IV. IMPLEMENTACIÓN

GWT funciona por módulos identificados por un archivo xml con nombre único, estos módulos son conjuntos de paquetes Java seleccionados para compilar. En nuestro caso

cada módulo es una capa de la adaptación.

Las tres capas se organizan en tres proyectos de este modo:

- Web Publish Project (ver Sección IV-C): Aplicación GWT Web dedicada a generar el archivo final, tiene la clase Entry point que es la primera en ser ejecutada cuando la página web cargue y donde se publicara la librería portada en el objeto JS window de la página web.
- Core-Facade Project (ver Sección IV-B): Aplicación GWT Java que contiene las capas Core y Facade.
- Back Project (ver Sección IV-A): Aplicación GWT Java que contiene la capa Back de la arquitectura.

El objetivo de la capa Facade es servir de interfaz a la capa Core, es decir que estarán fuertemente acopladas y por eso se dejan en el mismo proyecto. La capa Back con el Processing Stub puede ser reutilizada por otras adaptaciones, por eso se deja en un proyecto separado. La organización de los proyectos se ve en la Figura 3.

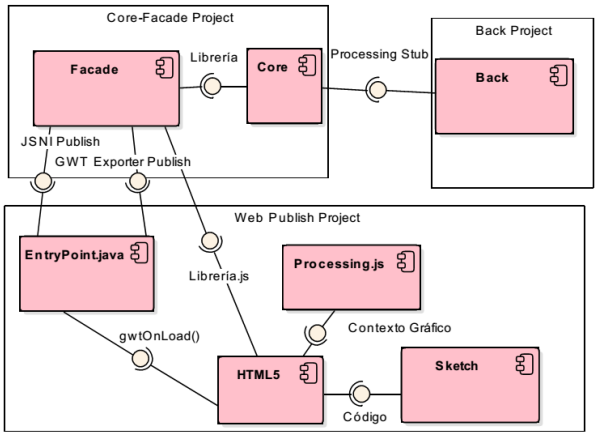


Figura 3. Implementación

A. *Proyecto Back: Processing Stub*

El propósito de esta capa es evitar la modificación del código original de la librería cuando realice llamados a Processing. Siendo PApplet la clase principal de Processing, crearemos un módulo GWT llamado Processing con una clase Stub que será llamada PApplet también. Esta encapsula llamados a atributos, métodos y funciones de Processing.js; emulando los métodos y el comportamiento de la clase PApplet original.

En esta clase Processing Stub se mantiene una referencia GWT JavaScriptObject al contexto gráfico de Processing.js (CGPJS en adelante); los atributos, métodos y funciones son accedidos desde Java a JS por medio de la JavaScript Native Interface (JSNI) [14]. Ver algoritmo 1 y la Figura 4.

Algoritmo 1 Clase Stub

```
public class PApplet {                                              1
 private JavaScriptObject context;                                  2
 public object atribute;                                            3
                                                                    4
 public PApplet (JavaScriptObject object){                          5
  init(object);                                                     6
 }                                                                  7
 private native void init(JavaScriptObject object)/*-{              8
  this.@processing.core.PApplet::context = object;                  9
  this.@processing.core.PApplet::atribute = object.atribute         10
      ; }-*/;
                                                                    11
 public final native void method (object o)/*-{                     12
  var context = this.@processing.core.PApplet::context;            13
  context.method(o); }-*/;                                          14
}                                                                   15
```

Línea2 Atributo context de tipo GWT JavaScriptObject, es una referencia al CGPJS, con el cual podemos acceder a objetos JS desde Java y viceversa usando JSNI.

Línea8 Método JSNI que asigna el CGPJS a la clase, conociendo el nombre de los atributos en JS, pueden ser asignados a

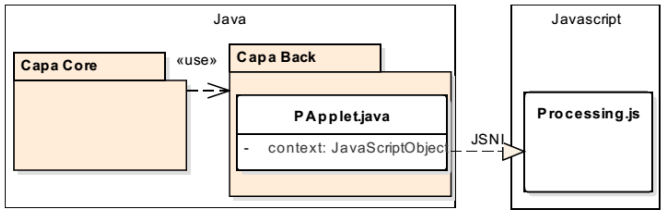


Figura 4. Estructura del Processing Stub

B. *Proyecto Core-Facade*

La capa Core contiene la librería original sin modificaciones. La capa Facade contiene la clase ExporterFacade.java, que es el wrapper JS de la librería compilada, en esta clase los objetos Java de la capa Core son recreados y publicados dentro de objetos JS utilizando JSNI. Estos objetos JS tienen el mismo nombre y comportamientos que sus originales en Java. Cada objeto JS mantiene una referencia a su objeto Java correspondiente para acceder a él desde la página web. Estos objetos JS pueden tener una referencia al CGPJS, de manera que puede ser usado luego en la clase Processing Stub en la capa Back de jerarquía inferior. Ver algoritmo 2.

Este tipo de publicación manual JSNI, dependiendo de la complejidad de la librería puede resultar largo y difícil, sin embargo existe una herramienta que realiza este proceso automáticamente: GWT EXPORTER [15].

Gwt Exporter: Esta librería GWT publica objetos Java en objetos JS igual que la publicación manual JSNI, pero usando anotaciones e interfaces. En este artículo, se mostrara como usar ambas opciones, ver Figura 3. Para que la librería original no sea modificada se utiliza la interfaz ExportOverlay. Ver algoritmo 3.

Algoritmo 2 Clase JsniFacade

```
1  public class ExporterFacade {
2
3   public static LibraryClass CreateLibraryClass(
        JavaScriptObject context,object param)
4       {return new LibraryClass(new PApplet(context),param);}
5
6   public  native void JsniPublish() /*-{
7      $wnd.LibraryClass  = function(context,param)
8    {var JSClass = @facade.client.JsniFacade::LibraryClass(
         Lcom/google/gwt/corepis /client/JavaScriptObject;FFF)
         (context,param);
9     JSClass.method = JSClass.@LibraryPackage.LibraryClass::
          method(F);
10    return JSClass;
11      }
12  }-*/;
13 }
```

Línea4 Instancia de la librería original, con algún parámetro requerido o el CGPJS para ser usando en la clase PApplet de la capa Back.

Línea7 Un objeto facade con el mismo nombre de la librería original es creado en el objeto $wnd, que es una constante de GWT para referirse al objeto window en JS, objeto principal JS de cada página html.

Línea8 Una vez el objeto JSClass JS ha sido instanciado, siendo JSClass una referencia a la clase Java, se pueden crear apuntadores JS a los métodos Java como en la línea 9.

C. *Web Publish Project*

Este es el proyecto principal que utiliza el usuario, aquí está contenida la página web html donde se muestra el sketch y se compila la capa Facade a JS. Todo proyecto web GWT debe tener una clase que implemente la interfaz GWT EntryPoint, en este caso Web.java. Esta interfaz GWT EntryPoint obliga a implementar el método onModuleLoad (), este método compilara a JS la clase ExporterFacade como se ve en la Figura 5

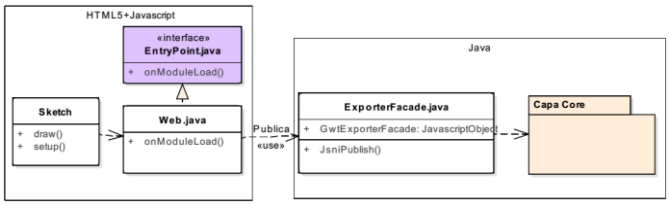


Figura 5. Proyecto Web Publish

El método onModuleLoad() es el primero en ejecutarse cuando se carga la página web. Processing.js debe iniciarse después de que publique la clase JsniFacade con GWT Exporter o manualmente con JSNI, ver el algoritmo 4.
Finalmente la librería es llamada en la página web como se ve en el algoritmo 5.

V. DESARROLLADORES JAVA

Hasta ahora el método propuesto está dirigido principalmente a los desarrolladores web JS; sin embargo nuestro método permite fácilmente adaptarse a los desarrolladores Java sin requerir casi ningún conocimiento JS; en lugar de exportar cada clase (y sus métodos), podemos crear una única clase Sketch.java (Ver Algoritmo 6), que contendrá todo el código del sketch. Esta clase Sketch.java debe heredar de la clase PApplet.java todos los métodos necesarios para funcionar y puede iniciarse con el CGPJS. Al final solo la clase Sketch.java y los métodos utilizados de Processing.js serán exportados usando GWT Exporter (Ver algoritmo 7), así todo el código será ejecutado de manera trasparente dentro de la clase portada Sketch.java en JS.

Algoritmo 3 Exportar con Gwt Exporter

```
public  class ExporterFacade {                                  1
 @ExportPackage("")                                             2
 @Export("LibraryClass")                                        3
 public static abstract  class GwtExporterFacade implements     4
       ExportOverlay<LibraryClass>
 {                                                              5
  @ExportConstructor                                            6
  public static LibraryClass constructor( JavaScriptObject     7
       context,object param){
   return new LibraryClass(new PApplet(context),param); }       8
  public abstract OtherLibraryClass method();                   9
  public abstract void method2(OtherLibraryClass other);        10
 },,                                                            11
 @ExportPackage("")                                             12
 @Export("OtherLibraryClass")                                   13
 @ExportClosure()                                               14
 public abstract class OtherLibraryClass implements             15
      ExportOverlay<OtherLibraryClass>   {
  public abstract int attribute();                              16
 }                                                              17
}                                                               18
```

Línea2 @ExportPackage: Da nombre al objeto JS que contendrá todas las clases exportadas, el objeto window JS es la selección por defecto.

Línea3 @Export: Marca una clase y métodos públicos para ser exportados. Define el nombre final de la clase o método en JS.

Línea6 @ExportConstructor: Marca un método que hará de constructor de la clase a exportar, para que esta etiqueta funcione, debe existir un constructor sin parámetros en la clase a exportar.

Línea14@ExportClosure: Marca una clase para ser un argumento de algún método exportado a JS.

Algoritmo 4 Implementando Gwt EntryPoint

```
public class Web implements EntryPoint {                       1
 public void onModuleLoad() {                                   2
  JsniFacade f = new JsniFacade(); //JSNI Export                3
  f.publish();                                                  4
  ExporterUtil.exportAll(); // Gwt Export                       5
  onLoadImpl();                                                 6
 }                                                              7
 private native void onLoadImpl() /*-{                          8
  if ($wnd.gwtOnLoad && typeof $wnd.gwtOnLoad == 'function      9
      ')
   $wnd.gwtOnLoad(); }-*/;                                      10
}                                                               11
```

Línea3 Publicando con JSNI.
Línea5 Publicando con GWT Exporter.
Línea9 Método JS gwtOnLoad.

VI. RESULTADOS

Para ilustrar nuestra propuesta, estas librerías fueron adaptadas exitosamente a la web manteniendo sus comportamientos:

- Obsessive Camera Direction (OCD) , [16] (ver Sección VI-A): Librería para el control de cámara de una escena, consta únicamente de la clase Camera.java, se adaptó con el propósito de probar la publicación JSNI y la capa Back, ya que internamente llama a la clase PApplet.java de Processing. Se adaptó el siguiente ejemplo: OCD Reference Zoom.

Algoritmo 5 Llamado de la librería desde la página web

```
1  <script type="text/JS" language="JS" src="webpublish/
       webpublish.nocache.js"></script>
2  <script src="processing.js"></script>
3  <script language='javascript'>
4   function gwtOnLoad()
5   { var p = new Processing("processing-canvas", sketchProc);
6     function sketchProc(processing) {
7     var c;
8     processing.setup = function() {
9      c = new LibraryClass(this, 1);
10    };
11    processing.draw = function()
12    {c.method(1);};
13   }
14  }
15 </script>
```

Línea9 Llamado de la librería. Se puede pasarle el CGPJS al constructor.
Línea12Llamado un método de la librería.

Algoritmo 6 Case Java con el código del *Sketch*

```
1  public class Sketch extends PApplet {
2   public Sketch(){}
3   public Sketch(JavaScriptObject ctx) {
4    super(ctx);
5   }
6   public void setup() {
7    size(100, 100);
8   }
9   public void draw() {
10   background(0);
11  }
12 }
```

Line2 Constructor vacío para que @ExportConstructor funcione.
Line3 Constructor donde la superclase PApplet.java es instanciada con el CGPJS.

- Traer's physics Port [17] (ver Sección VI-B): Librería de física que calcula y aplica colisiones, sistemas de partículas, etcétera, a objetos manipulables con el mouse.
  Se adaptó con el propósito de probar la publicación con GWT de una librería con varias clases; pero no realizan llamados a Processing.

Algoritmo 7 Usando la clase portada Sketch.java

```
1  <script type="text/JS" language="JS" src="webpublish/
       webpublish.nocache.js"></script>
2  <script src="processing.js"></script>
3  <script language='javascript'>
4   function gwtOnLoad()
5   { var p = new Processing("processing-canvas", sketchProc);
6     function sketchProc(processing) {
7     var c;
8     processing.setup = function() {
9      c = new Sketch( this );
10     c.setup();
11    };
12    processing.draw = function()
13    {c.draw();};
14   }
15  }
16 </script>
```

Línea9 Instancia la super clase Sketch.java y ejecuta setup().
Línea13Ejecuta draw().

Para demostrar la flexibilidad del método, las dos librerías fueron adaptadas utilizando el archivo de código .PDE de Processing y también con código JS embebido dentro de la página web, pueden verse corriendo junto con su código fuente en [18].

A. *Web ocd*

Empleando nuestro método, se crearon tres proyectos: la capa back Processing Stub, la capa core OcdJs y la capa facadeWebOcd que contienen tres módulos GWT, processing.gwt.xml, ocd.gwt.xml y facade.gwt.xml.

Se identifican los métodos que OCD utiliza de Processing son: PApplet.perspective y PApplet.camera. Así que los métodos Stub correspondientes fueron creados en la capa Back.

En la capa Core se coloca la librería OCD sin modificar. En la capa Facade se realiza la publicación en JS usando JSNI manualmente, exportando la clase Camera.java y sus métodos.

B. *Web Traer Physics y GWT Exporter*

Traer Physics 3.0 [19], no realiza llamados a Processing; pero las clases son complejas e implementan interfaces. Siguiendo el método, se crearon dos proyectos: la capa Core TraerPhy- sicsJs y la capa Facade WebTraerPhysics; contienen los dos módulos GWT: traerphysics.gwt.xml y facade.gwt.xml.

La capa Core contendrá la librería Traer Physics intacta y las clases principales para exportar son ParticleSystem.java, Vector3D.java y Particle.java.

La publicación a JS se realiza en la capa facade con la clase ExporterFacade.java usando GWT Exporter, allí se utilizó la anotación @ExporterPackge ("TraerPhysics"), lo cual significa que las clases están contenidas en el objeto JS TraerPhysics. Se utilizó la anotación @ExporterConstructor en la clase Particle.java, y se creó un constructor sin parámetros para ella.

C. *Ejemplos para desarrolladores Java y JavaScript*

Algunos sketches simples de Processing con interacción teclado y mouse han sido adaptados para probar los modos de desarrollo para programadores Java o JS; los cambios al código original fueron mínimos y también pueden verse corriendo juto a su código fuente en [18].

- Action Driven Callback: Sketch para desarrolladores Java donde se pueden arrastrar círculos con el Mouse. Se reutiliza la capa Back del ejemplo VI-A. El código del Sketch se mantiene intacto en la capa Core, sin embargo en la capa Facade, no se exportan los componentes del sketch si no únicamente la clase principal llamada ActionDrivenCallback.java y los métodos setup() y draw(); de manera que el sketch se mantiene completamente en Java; pero para modificar el código del sketch se requiere compilar el código de Java a JS.
- Boring clic And Drag: Sketch simple para desarrolladores JS donde se puede cambiar al azar la ubicación de círculos con un clic. Se reutiliza de nuevo la capa Back del ejemplo VI-A; pero la capa Core contiene únicamente los componentes del sketch: MouseAgent.java, TerseHandler.java y GrabbableCircle. En la capa Facade los componentes son exportados cada uno a un objeto JS conrrespondiente usando GWT Exporter. El código del sketch es escrito en JS y embebido en la página html, donde puede ser modificado sin necesidad de compilar a JS.

D. *Comparación y ventajas*

Para el desarrollador de librerías, las principales ventajas que ofrece el método son dos: permite tener un solo trunk de desarrollo y no se requiere tener conocimientos en JS, sin embargo es flexible para quien quiera usar JS o Java.

El enfoque por módulos permite la reutilización de código, por ejemplo la capa Back que contiene el Processing Stub, puede adaptarse para ser utilizado por otras librerías que lo requieran, incluso esta capa puede contener otros motores gráficos. El enfoque permite también crear diferentes módulos para varias utilidades que pueden ser recurrentes en futuras adaptaciones como hilos, tablas Hash, parsers de arreglos entre Java y JS, utilidades matemáticas, acceso a las Apis de HTML5, etcétera.

VII. Conclusiones

Se ha presentado un método para adaptar librerías de Processing a Processing.js, sin conocimientos de JS y manteniendo un solo trunk de desarrollo. Las librerías adaptadas en este artículo son solo pruebas de concepto que demuestran que desarrollando más el método podremos en el futuro adaptar librerías con mucha complejidad como Proscene [20].

Como trabajo futuro, se podría automatizar este método en una herramienta para el IDE de Processing o para Eclipse IDE; sin embargo las limitaciones del método están dadas por el uso de GWT, diferencias irreconciliables entre los lenguajes Java y JS como, hilos, reflection y demás; pero pueden ser abordados con nuevas tecnologías como HTML5 Web Workers and extensions.

Referencias

[1] C. Reas and B. Fry, Getting Started with Processing. O'Reilly, 2010.

[2] L. Burdy, A. Requet, and J.-L. Lanet, "Java applet correctness: A developer-oriented approach," in FME 2003: Formal Methods. Springer, 2003, pp. 422–439.

[3] Adobe, "Adobe flash platform," http://www.adobe.com/flashplatform/, 2012. [Online]. Available: \protect\unhbox\voidb@x\penalty\@M\http://www.adobe.com/flashplatform/

[4] Microsoft, "Microsoft silverlight perspective 3d graphics," 2012. [Online]. Available: http://www.microsoft.com/silverlight/perspective-3d-graphics/

[5] D. Brutzman and L. Daly, X3D: extensible 3D graphics for Web authors. Morgan Kaufmann, 2010.

[6] J. Resig, B. Fry, and C. Reas, "Processing. js," 2012.

[7] T. Parisi, WebGL: Up and Running. O'Reilly Media, 2012. [Online]. Available: http://shop.oreilly.com/product/0636920024729.do

[8] J. Vantomme, Processing 2: Creative Programming Cookbook: Over 90 Highly-effective Recipes to Unleash Your Creativity with Interactive Art, Graphics, Computer Vision, 3D, and More. Packt Publishing, 2012, chapter 9: Exploring JavaScript Mode.

[9] P. team, "Processingjs exhibition," 2015. [Online]. Available: http://processingjs.org/exhibition/

[10] K. Phillips, "Toxiclibs.js open source computational design," http://haptic-data.com/toxiclibsjs/, 2011. [Online]. Available: http://labs.hapticdata.com/2011/01/toxiclibs-js-open-source-computational-design/

[11] Google, "Understanding the gwt compiler," https://developers.google.com/web-toolkit/doc/latest/DevGuideCompilingAndDebugging, October 2012. [Online]. Available: https://developers.google.com/web-toolkit/doc/latest/DevGuideCompilingAndDebugging#DevGuideJavaToJavaScriptCompiler

[12] ——, "Jre emulation reference," https://developers.google.com/web-toolkit/doc/latest/RefJreEmulation, October 2012. [Online]. Available: https://developers.google.com/web-toolkit/doc/latest/RefJreEmulation

[13] Oracle, "Stubs and skeletons," 2010. [Online]. Available: http://docs.oracle.com/javase/7/docs/platform/rmi/spec/rmi-arch2.html

[14] Google, "Coding basics - javascript native interface (jsni)," https://developers.google.com/web-toolkit/doc/latest/DevGuideCodingBasicsJSNI, 2012. [Online]. Available: https://developers.google.com/web-toolkit/doc/latest/DevGuideCodingBasicsJSNI

[15] M. C. M. Ray Cromwell, "gwt-exporter," http://code.google.com/p/gwt-exporter/, 2012. [Online]. Available: http://code.google.com/p/gwt-exporter/

[16] K. L. Damkjer, "Obsessive camera direction (ocd) reference," http://www.gdsstudios.com/processing/libraries/ocd/reference/, 2009. [Online]. Available: http://www.gdsstudios.com/processing/libraries/ocd/reference/

[17] M. Niemi, "Traer's physics library to processing.js - notes," http://svbreakaway.info/tp.php, 2011. [Online]. Available: http://svbreakaway.info/tp.php#tpjs

[18] X, "processing adapted libraries," http://goo.gl/KzvxH1, 2013. [Online]. Available: http://goo.gl/KzvxH1

[19] J. T. Bernstein, "Traer.physics 3.0," http://murderandcreate.com/physics/, 2010. [Online]. Available: http://murderandcreate.com/physics/

[20] J. P. Charalambos, "Proscene description," http://code.google.com/p/proscene/, 2011. [Online]. Available: http://code.google.com/p/proscene/

Cesar Colorado nació en la ciudad de Bogotá en 1988. Se gradúa con el título de Ingeniero de Sistemas en la Universidad Nacional de Colombia - Sede Bogotá en 2011.
De 2010 a 2015, ha ejercido desarrollando aplicaciones web para el sector financiero. Desde 2010, ha estado en el grupo de investigación sobre computación gráfica RemixLab, perteneciente a la Facultad de Ingeniería de la Universidad Nacional de Colombia. En esta misma institución, es candidato actualmente al título de "Magister en Ingeniaría – Ingeniería de Sistemas y computación". Sus temas de interés para investigación incluyen gráficos para la web, realidad aumentada y gráficos 3d.

Jean Pierre Charalambos recibe en 1994 el título de Ingeniero Industrial de la Pontificia Universidad Javeriana Sede Bogotá. Recibe el título de "Magister en Ingeniería de Sistemas" en la Universidad Nacional de Colombia - Sede Bogotá y luego realiza un Doctorado en Software en la Universidad Politécnica de Cataluña que termina en 2008.
Ha ejercido como docente en la Pontificia Universidad Javeriana y la Universidad Nacional de Colombia , donde funda y dirige el grupo de investigación sobre computación gráfica RemixLab. Ha publicado varios artículos, asistidito a varios eventos científicos y ha sido director de varias tesis de postgrado, además de desarrollar el software para control de cámara Proscene. Sus temas de interés para investigación incluyen rendering en tiempo real, visualización científica, grádicos 3d y la interacción hombre-máquina.
El doctor Charalambos recibió un reconocimiento por parte de Qtopia Worldwide Developer Contest,Trolltech y Sharp Co en 2002 y ha sido miembro del grupo de investigación Bioengenium así como del Computer graphics Institute de la Vienna University of Technology.

Framework Basado en ODA para la Descripción y Composición de Servicios Web Semánticos (FODAS-WS)
ODA Based Framework for Description and Composition of Semantic Web Services (FODAS-WS)

Jose Aguilar y Omar Portilla

***Resumen*— Este artículo presenta el Framework FODAS-WS, basada en la arquitectura ODA (Ontology Driven Architecture). FODAS-WS es un marco de referencia que sirve para diseñar y describir Servicios Web Semánticos (SWS) automáticamente. Con FODAS-WS se busca facilitar las tareas de descubrimiento, composición y ejecución automática de SWS. Para describir un servicio se utilizan tres capas de abstracción: CAPACIM, CAPAPIM y CAPAPSM. La CAPACIM abarca una descripción semántica del dominio y de las funcionalidades del servicio, y es descrita manualmente por el diseñador. Las capas CAPAPIM y CAPAPSM son generadas por procesos de transformación automáticos, basados en las capas que le siguen en mayor nivel de abstracción. CAPAPIM usa como insumo la salida de CAPACIM, y CAPAPSM la salida de CAPAPIM. CAPAPIM tiene un modelo ontológico para generar el esqueleto del código del servicio, en el cual se estipulan los métodos y parámetros que el servicio requiere. CAPAPSM especifica la composición y ejecución automática de servicios.**

***Palabras clave*— Servicios Web Semánticos, ODA Diseño de SWS.**

***Abstract*—This article presents the FODAS-WS Framework, based on the ODA architecture (Ontology Driven Architecture). FODAS-WS is a framework used to design and describe Semantic Web Services (SWS) automatically. FODAS-WS facilitates the tasks discovery, the composition and automatic execution of SWS. To describe a service are used three layers of abstraction: CAPACIM, CAPAPIM and CAPAPSM. The CAPACIM covers a semantic description of the domain and the functionality of the service, and is described by the designer manually. The CAPAPIM and CAPAPSM layers are generated by automated transformation processes based on the layers that follow a greater level of abstraction. CAPAPIM uses as input the CAPACIM output, and CAPAPSM the CAPAPIM output. CAPAPIM has an ontological model to generate the skeleton of the service code, in which are defined the methods and parameters set required by the service. CAPAPSM automatically specifies the composition and execution of services.**

***Index Terms*—Semantic Web Services, ODA, Design SWS.**

I. Introducción

Según [1], la globalización, cooperación y colaboración presentadas en la actualidad han cambiado substancialmente el mundo del software. Actualmente, un usuario humano de forma individual no puede producir conocimiento y ser competente. Pero, en colaboración con otros y con organizaciones, puede constituir una rica fuente de conocimiento y competencias. El conocimiento puede ser usado, almacenado, descubierto e intercambiado por cualquier usuario (humano, dispositivo industrial inteligente, robot, agente software, etc.) en ambientes heterogéneos y amplios.

En la actualidad se cuenta con infraestructuras de computación distribuida adecuadas para la integración de datos. Los servicios web se constituyen como los principales representantes de tal infraestructura tecnológica, y se caracterizan por su tecnología abierta, dinámica y con bajo nivel de acoplamiento. Estas características proveen un buen soporte para compartir recursos y realizar trabajos cooperativos en ambientes heterogéneos. Adicional a esta tecnología se encuentra la web semántica, que representa una gran revolución en la forma de almacenar y describir conocimiento a través del uso de ontologías. Las ontologías juegan un papel fundamental

Dr. Jose Aguilar ha sido parcialmente financiado por el Proyecto Prometeo del Ministerio de Educación Superior, Ciencia, Tecnología e Innovación de la República de Ecuador).

J. A. está con la Universidad de Los Andes, Facultad de Ingeniería, Escuela de Ingeniería de Sistemas, CEMISID, Mérida, Estado Mérida, Venezuela, Tlf. +58-274-2402880, Fax: +58-274-2402872, E-mail: aguilar@ula.ve. Además, J. A. actualmente es Investigador Prometeo en la Universidad Técnica Particular de Loja, Loja, Ecuador,

Omar Portilla es profesor del programa de Ingeniería de Sistemas de la Universidad de Pamplona, Colombia. Está culminando el Postgrado de Computación, Facultad de Ingeniería, Universidad de Los Andes, Mérida, Estado Mérida.

en la interoperabilidad, ya que son formas estructuradas para describir y formalizar los vocabularios necesarios para compartir conceptualizaciones, con lo que contribuyen a resolver la heterogeneidad semántica, proveyendo una conceptualización compartida en cierto dominio de interés.

Este artículo propone el uso de ontologías organizadas mediante la arquitectura ODA, para el diseño de SWS, y para la realización de descubrimiento, composición y ejecución automática de ellos. En las siguientes secciones se propone un Framework para la descripción de SWS basado en ODA, denominado FODAS-WS, que pretende servir como instrumento para realizar la generación de código, tanto del servicio web diseñado (esqueleto de implementación del SWS), como del cliente del SWS. Esto permite el descubrimiento, la ejecución y la composición automática de SWS.

II. MARCO TEÓRICO

A. Servicios Web Semánticos

De acuerdo a lo planteado en [2]: "Un servicio web es un componente de software modular bien definido que expone una interfaz sobre la red. El uso de los servicios web se hace a través del intercambio de mensajes XML a través del protocolo HTTP".

El objetivo del servicio web es soportar una infraestructura abierta para aplicaciones web. También, deben tener una descripción específica que determine su definición, para qué está hecho el servicio web, y cuáles servicios se pueden invocar.

Los SWS son considerados por [3], como una extensión a los servicios web, que proporcionan ventajas como el descubrimiento automático de servicios, la invocación automática del servicio, y la composición e interoperabilidad del servicio web.

B. Generación automática de código

Según [5], un generador de código automático es una herramienta que genera, a partir de determinados patrones, el código fuente de una aplicación. El uso de estas herramientas reduce el tiempo que se necesita para el desarrollo del software, como también asegura que el grado de errores de programación permanezcan acotados, reduciendo por tanto los tiempos de depuración y puesta a punto. Estos sistemas constan de procedimientos que generan el código fuente, en función de lo expresado en el diseño de la aplicación o modelo de datos.

Dentro del ámbito de la generación automática de código las propuestas más populares son las relativas a la implementación de soluciones a partir de diagramas UML [8]. En este enfoque, se intenta traducir los esquemas de relaciones entre clases que representan a una base de datos, a clases e interfaces implementadas en un lenguaje en particular, tal como lo plantean [6] y [9].

C. Arquitectura MDA y ODA

En [10] y [11] se propone una clasificación para entender cómo se aplican las ontologías en Ingeniería del Software, y cuál es el beneficio en cada caso. Primero, distinguen el rol de las ontologías en el contexto de ingeniería del software entre su uso en tiempo de ejecución y en tiempo de desarrollo. Segundo, miran el tipo de conocimiento que la ontología maneja. Además, distinguen entre el problema del dominio que el software aborda y los aspectos de infraestructura que hace el desarrollo de software más conveniente. Basados en estas dos dimensiones, en [10] plantean la matriz de la Figura 1, en las que se observan cuatro áreas básicas:

- Ontology-driven development (ODD): Inserta el uso de ontologías en el tiempo de desarrollo, para describir el problema del dominio. Por ejemplo, la arquitectura MDA propone ontologías para esta área.
- Ontology-enabled development (OED): También usa ontologías en tiempo de desarrollo, Pero para soportar las tareas realizadas por los desarrolladores. Por ejemplo, búsqueda de componentes o resolución de problemas de soporte.
- Ontology-based architectures (OBA): Usa las ontologías como artefacto primario en tiempo de ejecución. Acá, las ontologías forman parte fundamental de la lógica de la aplicación. Por ejemplo, aplicación con acceso a reglas de negocio.
- Ontology-enabled architectures (OEA): Finalmente, las ontologías se toman para proveer infraestructura de soporte en los sistemas de software durante el tiempo de ejecución. Un ejemplo son los SWS, donde las ontologías adicionan una capa semántica de mayor nivel que las descripciones clásicas de servicios web. En esa capa, se adiciona funcionalidades para el descubrimiento automático de servicios web, y emparejamiento y composición automática de servicios.

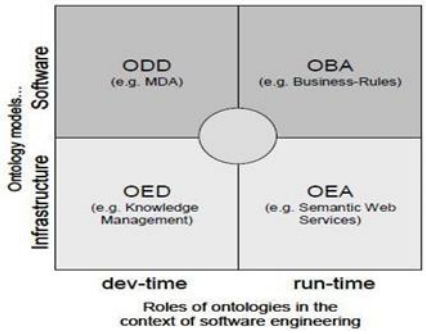


Figura 1. Uso de categorías para ontologías en ingeniería del software propuestas en [10].

Aunque las cuatro áreas parecen distintas en primera instancia, ellas se entrecruzan en algunas aplicaciones.

II.C.1) Ingeniería Manejada por Modelos (Model Driven Engineering)

Según [12], el objetivo de la Ingeniería Manejada por Modelos (MDE) es manejar el desarrollo de software basado en el modelado del dominio. En otras palabras, MDE se focaliza en la especificación de sistemas, más que en su implementación. MDA es un campo de la MDE, que especifica una metodología para el desarrollo de sistemas de software separando la lógica del negocio de la lógica de la plataforma tecnológica subyacente [13]. MDA especifica tres niveles en su arquitectura:

- Modelo Independiente de Computación (Computation Independent Model CIM): describe el contexto en el cual el sistema se usará.
- Modelo Independiente de Plataforma (Platform Independent Model PIM): Describe el sistema en si mismo, sin considerar detalles del uso de plataformas. Un PIM puede ser satisfecho por uno o varias plataformas arquitectónicas reales.
- Modelo Especifico de Plataforma (Representation of Platform Specific Model PSM): En este nivel, se toma en cuenta el ambiente de implementación y los lenguajes.

II.C.2) Ontology-Driven Architecture (ODA)

El grupo de la W3C Semantic and Web Best Practices and Deployment Working Group (SWBPD) propuso a ODA como una extensión a MDA, que usa ontologías para explotar las ventajas de las tecnologías de web semántica.

Las ontologías describen las propiedades, relaciones y comportamientos de los componentes requeridos en un proceso de desarrollo de software. Las ontologías son usadas como un modelo conceptual en el desarrollo de componentes de software, tanto en tiempo de diseño como en tiempo de ejecución. Así, las ontologías son divididas en módulos genéricos, para modelar tanto semántica como metadatos sintácticos.

ODA provee representación de vocabularios de dominios ambiguos (como por ejemplo: requerimientos, restricciones, servicios, propiedades, etc.), chequeo de consistencia de los modelos, así como habilita nuevas capacidades automáticas en ingeniería del software. Varios trabajos han sido desarrollados para ilustrar como ODA puede ser usado en diseño, desarrollo y administración de sistemas distribuidos.

De acuerdo con [13], ODA permite integrar las ventajas de las tecnologías de web semántica en la metodología MDA, en las siguientes áreas:

- Sistemas e Ingeniería del Software, para utilizar la Web Semántica como una herramienta para modelado semántico, durante la especificación y diseño del software a lo largo de su ciclo de vida. También puede ser usada para describir, identificar, descubrir, y almacenar artefactos, y diseñar equipos.
- Especificación de modelos formales, para el uso de ontologías y tecnologías de web semántica como un medio de comunicación formal entre agentes, humanos o no, que participen en el proceso de desarrollo de software.
- Soporte del ciclo de vida del software. Esta perspectiva propone el uso de ontologías, como proveedor de lenguaje, terminología y estándares, para la especificación del dominio durante el ciclo de vida del software.
- Definición de Contenidos reusables y metadatos, como un poderoso descriptor de servicios y componentes, buscando facilitar el descubrimiento de servicios basados en descripciones precisas.

D.ESTADO DEL ARTE

En [14] se presenta un framework ontológico estructurado, llamado Web Services Framework (WSF). Se trata de una plataforma basada en servicios, que utiliza lenguajes de descripción específicos para la web. Propone ontologías y modelamiento basado en ontologías para mejorar lo propuesto en UML. Este framework, contempla los servicios web semánticos y usa las ontologías para capturar las propiedades funcionales y no funcionales de los servicios. El WSF cuenta con ontologías para modelar las tres capas propuestas por MDA. Plantea un proceso de transformación de CIM a PIM, pero no lo implementa en forma automática.

En [15] se propone un Framework manejado por modelos, para el desarrollo de aplicaciones web orientadas a servicios. Este Framework presenta el Lenguaje de Modelamiento de Presentación (PML), haciendo particular énfasis en los procesos de transformación. El trabajo enfatiza en la fase especifica de generación de código.

En [16] proponen el MWSAF (METEOR-S Web Service Annotation Framework), es un Framework para marcación semiautomática de Servicios Web, con ontologías. En él se desarrollan algoritmos para emparejar y anotar archivos WSDL, con las ontologías propuestas.

En [17] se sigue y se adapta la metodología MDD, para su aplicación con el desarrollo de un tipo de software: Servicios web semánticos, basándose en la especificación OWL-S.

En [18] sintetizan los enfoques de desarrollo dirigido por modelos, aplicados al desarrollo de servicios web semánticos bajo dos categorías: los basados en principios y metodologías de Ingeniería de Software, y los basados en la aplicación formal de UML [19] y [20].

III. ARQUITECTURA DEL FRAMEWORK BASADO EN ODA PARA LA DESCRIPCIÓN Y COMPOSICIÓN DE SERVICIOS WEB SEMÁNTICOS (FODAS-WS)

El FODAS-WS contiene un conjunto estandarizado de conceptos, prácticas y criterios para diseñar SWS, el cual sirve como referencia para diseñar, implementar (generación de esqueletos de código del SW), descubrir automáticamente, emparejar y componer automáticamente (generación del código del cliente), SWS. FODAS-WS representa una herramienta para la realización de diseño e implementación de servicios web que usa ODA. Proporciona un elevado nivel de definición semántica, que puede soportar procesos de composición automática reflejados en la generación automática del código de la aplicación cliente.

FODAS-WS busca contribuir en el proceso de descubrimiento, emparejamiento y composición automática de servicios, mediante modelos ontológicos comprensibles a la máquina, almacenados en tres capas de distinto nivel de abstracción, que se llaman CAPACIM, CAPAPIM y CAPAPSM. Estas capas se ajustan completamente lo propuesto en la arquitectura ODA. Cada uno de los modelos ontológicos que constituyen las distintas capas de FODAS-WS, se describen en lenguaje OWL, el cual le aporta no solo sintaxis, sino semántica y métodos formales de análisis, inferencia y razonamiento sobre el conocimiento expresado.

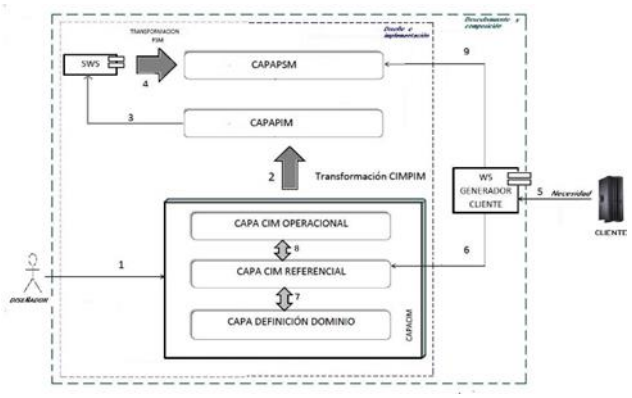


Figura 2. Arquitectura de FODAS-WS

FODAS-WS es implementado en java, como una estructura conceptual y tecnológica, que sirve de base para el diseño, implementación, y descripción de SWS, así como para la generación automática de código del cliente del servicio. La Figura 2 ilustra la arquitectura propuesta para FODAS-WS y sus distintos componentes e interrelaciones.

La arquitectura de FODAS-WS posee siete componentes que interactúan entre sí, los cuales son:

- *CAPACIM:* Representa la capa de mayor abstracción en la arquitectura ODA. Se compone de tres modelos ontológicos, en los que se modela el dominio sobre el cual el servicio web opera, y aspectos de diseño relacionados con el proceso que se implementa en el servicio, como por ejemplo: funcionalidades que provee el servicio, descripción del flujo de actividades que conforman el proceso, y descripción de la dinámica e interacción que se generan entre los objetos participantes en el proceso. Además, está capa se encarga de describir el modelo arquitectónico del servicio implementado, y la definición semántica de las capacidades que ofrece, lo cual sirve como insumo para la realización del proceso de descubrimiento automático y de emparejamiento.
- *CAPAPIM:* Esta capa contiene un modelo ontológico, que se genera de forma automática a partir de lo descrito en la CAPACIM, mediante un proceso de transformación. En este nivel de abstracción se plasma lo necesario para generar el esqueleto de código de la implementación del servicio, en el cual se estipulan los métodos y parámetros que el servicio requiere, así como la secuencia de llamadas de los distintos métodos. Esta capa posibilita la generación automática del esqueleto de código del servicio, además de disminuir el trabajo de implementación, garantizando total compatibilidad entre lo implementado con lo diseñado, de tal forma que la composición y ejecución automática se realice con total compatibilidad.
- *CAPAPSM:* Es la capa más específica con que cuenta FODAS-WS. En ella se estipula lo necesario para realizar la composición y ejecución automática de servicios. Esta capa describe semánticamente lo estipulado en un archivo WSDL, fusionado con lo contemplado en la especificación OWL-S . Esta capa se genera de forma automática, una vez implementado y desplegado el servicio. Con ello, se busca que la información registrada en lo asociado con PSM, sea totalmente compatible con los detalles de implementación.
- *TRANSFORMACIÓN CIMPIM:* permite realizar la generación automática de la CAPAPIM, a partir de lo descrito semánticamente en la CAPACIM.
- *TRANSFORMACIÓN PSM:* permite realizar la generación automática de la CAPAPSM, a partir de lo descrito semánticamente en la CAPAPIM. Es importante resaltar que dicho proceso de transformación, se realiza después de implementado y desplegado el servicio web.
- *SWS:* Es el SWS diseñado, implementado y desplegado.
- *WS GENERADOR CLIENTE:* Es un servicio web propuesto para ser implementado en trabajos futuros. Requiere como entradas las necesidades del cliente, expresadas en forma semántica, de acuerdo con las políticas semánticas de FODAS-WS. WS GENERADOR CLIENTE se encarga de realizar descubrimiento automático de los distintos SWS diseñados usando FODAS-WS, que emparejen con las necesidades solicitadas por la aplicación. Una vez descubierto todos los servicios candidatos a ser seleccionados, el WS GENERADOR CLIENTE, mediante la aplicación de mecanismos de emparejamiento, selecciona el servicio que mejor empareja. Posteriormente, genera código automático que implementa al cliente para el SW seleccionado, realizando la ejecución automática del servicio, retornando a la aplicación solicitante los resultados arrojados en el formato que se infirió a partir de las necesidades descritas semánticamente que proporcionó.

A continuación se presenta la secuencia de interacciones que se generan entre los distintos componentes que intervienen en la arquitectura de FODAS-WS, para los aspectos de diseño e implementación necesarios:

1. El diseñador describe semánticamente la CAPACIM, en la cual contempla todo lo necesario para la especificación total del servicio.
2. CIMPIM genera la CAPAPIM automáticamente, a partir de la CAPACIM.
3. Una vez generada la CAPAPIM, FODAS-WS genera el código automático, que contiene el esqueleto del SWS.
4. Al momento en que se encuentre implementado y desplegado el SWS, la transformación PSM se encarga de generar de forma automática la CAPAPSM.

A continuación se describen la secuencia de interacciones que se dan, cuando se pretende hacer descubrimiento, composición y ejecución automática de servicios web:

1. Una aplicación cliente solicita a WS GENERADOR CLIENTE que le realice el descubrimiento, composición y ejecución automática de un servicio web que satisfaga las necesidades descritas semánticamente, y le retorne los resultados deseados.
2. WS GENERADOR CLIENTE acude a la CAPA CIM REFERENCIAL para obtener todo el conocimiento necesario para realizar el proceso de emparejamiento entre las

capacidades ofrecidas por los servicios y las necesidades solicitadas por la aplicación cliente.

3. La capa CIM REFERENCIAL accesa la CAPA DE DEFINICION DE DOMINIO, para realizar todo el proceso de inferencia que se requiera como insumo, en el momento de razonar sobre el diseño del SWS almacenado en la CAPA CIM OPERACIONAL.

4. La capa CIM REFERENCIAL accesa la CAPA CIM OPERACIONAL, para realizar el proceso de descubrimiento automático, usando como insumos las inferencias realizadas en el paso 3.

5. WS GENERADOR CLIENTE acude a la CAPA PSM, para obtener la información requerida para realizar composición y ejecución automática, usando como insumos las inferencias realizadas en los pasos 3 y 4. Una vez ejecutado el SWS, el WS GENERADOR CLIENTE retorna a la aplicación cliente, en el formato adecuado de acuerdo a lo inferido de las necesidades solicitadas, los resultados.

IV. DISEÑO, IMPLEMENTACIÓN Y PRUEBAS DE FODAS-WS

El caso de estudio presentado, consiste en el diseño e implementación de un servicio web semántico que básicamente es un sistema recomendador. Según [21] y [22], los sistemas RECOMENDADORES son SOFTWARE que ayudan a emparejar a USUARIOS con PRODUCTOS. Se consideran como agentes software que contienen los intereses individuales y particulares de un consumidor, y hacen recomendaciones de acuerdo a ello. Ellos proveen sugerencias de calidad a un consumidor, en el instante en que requiera buscar y seleccionar algún tipo de producto online. De esta manera, se obtiene un tipo de recomendación personalizada de acuerdo con las características y necesidades de los usuarios. El sistema recomendador del caso de estudio está en capacidad de recomendar dos tipos de productos: Objetos de aprendizaje y documentos web. Los metadatos de esos productos se encuentran almacenados en repositorios semánticos de objetos de aprendizaje y en repositorios semánticos de archivos web respectivamente.

A. DISEÑO E IMPLEMENTACIÓN DE LA CAPA CIM

Como ya se mencionó, la CAPACIM se compone de tres subcapas, cada una asociada con un modelo ontológico. Las capas se conocen como CAPA DE DEFINICIÓN DEL DOMINIO, en la que se modela el dominio sobre el cual el servicio web opera; CAPA CIM OPERACIONAL, en la cual se definen los aspectos de diseño relacionados con el proceso que se implementa en el servicio; y CAPA CIM REFERENCIAL, que se encarga de describir el modelo arquitectónico del servicio implementado, y la definición semántica de las capacidades que ofrece, lo cual sirve como insumo para la realización del proceso de descubrimiento automático y de emparejamiento.

IV.A.1) CAPA DE DEFINICIÓN DEL DOMINIO

En la CAPA DE DEFINICIÓN DEL DOMINIO, se proponen algunas definiciones:

- DOMINIO: Se conforma de un conjunto de ELEMENTOS que, mediante ASOCIACIONES, se relacionan entre ellos a través de la ejecución de ACCIONES.
- DEFINICIÓN SEMÁNTICA DEL DOMINIO: Es un conjunto de ASOCIACIONES, acompañado de una DEFINICIÓN SEMÁNTICA DE ACCIONES, y una DEFINICIÓN SEMÁNTICA DE ELEMENTOS.
- DEFINICIÓN SEMÁNTICA DE ACCIONES: Consiste en el establecimiento de relaciones semánticas entre las distintas ACCIONES que existen. Se cuentan con 8 ocho tipos de relaciones posibles entre las acciones, tal como se muestra en la Tabla 1.

RELACIÓN	REL. INVERSA	DESCRIPCIÓN
complementaA	complementadaPor	Una acción complementa a otra cuando al realizarla logra realizar tareas complementarias. Es decir, realiza aspectos que la otra tarea no lograba realizar
esSubAccion	esSuperAccion	Establece una jerarquía de acciones
especializaA	esEspecializadaPor	Determina cuando una acción genérica se realiza para un escenario especifico.
contrarioCon		Establece que acción realiza justo lo contrario que otra
genera		Determina cuando la realización de una acción deriva o genera otras acciones
semejanteA		Establece cuando dos acciones realizan tareas similares
profundizaA	esProfundizadoPor	Indica cuando una acción tiende a realizar lo mismo que otra pero en menor grado de profundidad.

Tabla 1. Especificación de Acciones.

-
- DEFINICIÓN SEMÁNTICA DE ELEMENTOS: Se encarga de determinar la jerarquía entre los elementos existentes, y de puntualizar los distintos elementos miembros de otro elemento (ver Tabla 2).

RELACIÓN	REL. INVERSA	DESCRIPCIÓN
esSubElemento	esSuperElemento	Determina la jerarquía entre elementos
tieneMiembro	esMiembroDe	Define cuales elementos son miembros de cierto elemento que los agrupa como conjunto con características similares.

Tabla 2. Definición de elementos.

- ASOCIACIÓN: La asociación es la encargada describir como un ELEMENTO, al ejecutar una ACCIÓN, genera otro elemento. Para describir una ASOCIACIÓN se usan seis relaciones, tal como se explica en la Tabla 3.

RELACIÓN	REL. INVERSA	DESCRIPCIÓN
tieneAntecedente	esAntecedenteDe	Esta relación determina cual es el elemento que ejecuta la

		acción dentro de la asociación definida.
tieneAccion	esAccionDe	Determina cual es la acción ejecutada dentro de la asociación definida
tieneConsecuente	esConsecuenteDe	Determina el elemento que se genera al ejecutar la acción

Tabla 3. Descripción de Asociaciones.

Además de definir ASOCIACIONES, ELEMENTOS y ACCIONES, también se especifican especializaciones de acciones, mediante las cuales se logra definir semánticamente los roles de cada elemento, y las formas en que son usadas en cada acción.

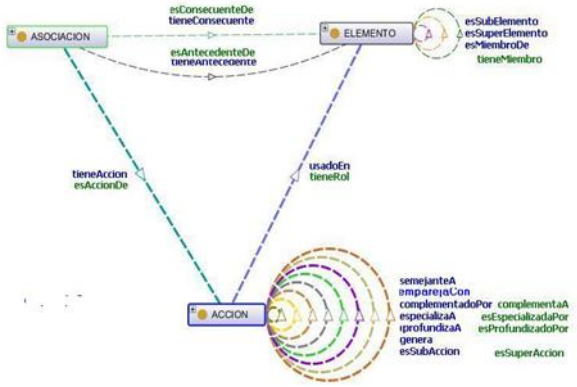


Figura 3. Modelo ontológico de la capa de definición del dominio.

En la Figura 3 se presenta el modelo ontológico usado para realizar la DEFINICIÓN SEMÁNTICA DEL DOMINIO. Como ya se explicó, la definición semántica del dominio consta de una definición semántica de acciones, y una definición semántica de elementos y un conjunto de asociaciones. El modelado de dominio para el caso de uso es mostrado en la Figura 4.

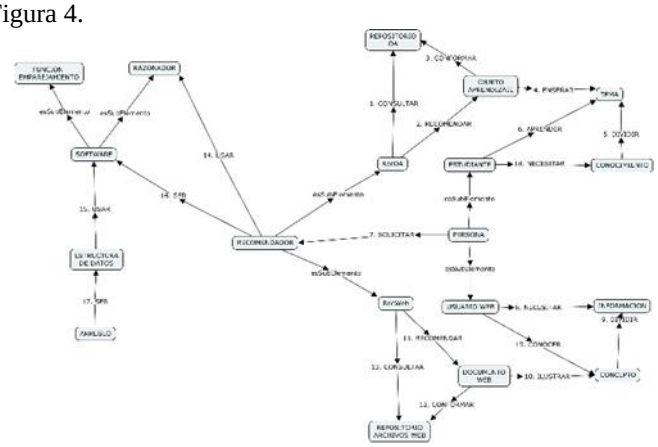


Figura 4. Ilustración de la definición semántica del dominio.

La Figura 4 muestra la representación gráfica de la definición semántica del domino de recomendación. Los metadatos de cada uno de estos dos tipos de productos a recomendar, se encuentran almacenados en un repositorio semántico conocidos como repositorioOA y repositorioWeb. Los óvalos en la Figura 4, representan los elementos definidos en el dominio, cada arco que une un par de óvalos representan una asociación. Cada asociación puede verse como una relación entre dos elementos, a través de una acción identificada con la etiqueta que acompaña a cada arco. Por ejemplo, la asociación1 determina que un elemento RecOA (Recomendador de Objetos de aprendizaje) ejecuta la acción consultar respecto del elemento RepositorioOA (Repositorio de Objetos de Aprendizaje).

La definición semántica de acciones en el caso de uso acá presentado, utiliza las acciones definidas en la Tabla 4, las cuales se registran en la ontología.

Relación	Parejas de acciones relacionadas
semejanteA	aprender- conocer
	Recomendar - sugerir
	Informar - ilustrar
	Requerir - necesitar
	Ordenar - priorizar
emparejaCon	Anteceder - seguir
	Enseñar- aprender
complementaA	Buscar - recomendar
	Consultar - recomendar
profundizaA	Recomendar - buscar
	Aprender - conocer
	Enseñar - ilustrar
genera	Recomendar - calcular
	Recomendar - ordenar
especializaA	ConsultarOA - consultar
	ConsultarWeb - consultar

Tabla 4. Definición de acciones para el dominio de recomendadores.

Por otro lado, ASOCIACIÓN se encarga de determinar cómo dos elementos se vinculan (asocian) mediante una acción. En la Figura 4 se observa la capa de definición semántica de dominio en la que se cuentan con 19 asociaciones, representadas como arcos que unen a los elementos que constituyen óvalos.

Es importante resaltar que la ASOCIACIÓN es fundamental, porque, es en torno a él que se da el proceso de descripción de las capacidades, los efectos, las necesidades, las precondiciones y los estados. Cada uno de ellos, definido como un conjunto de una o más asociaciones, que se usarán en los procesos de descubrimiento automático y emparejamiento.

IV.A.1.a) DISEÑO E IMPLEMENTACIÓN DE LA CAPA CIM REFERENCIAL

Es la capa en la que se contemplan los distintos referentes teóricos arquitectónicos y conceptuales a los que se ajusta el servicio web diseñado. Esta subcapa pretende aportar información relacionada con la forma en que está diseñado e implementado el servicio web, y las arquitecturas a las que se ajusta. Tal conocimiento se presenta para indicar a la máquina el modelo arquitectónico que se sigue, para que así ella use tal información en el momento de realizar el emparejamiento con las necesidades arquitectónicas que tenga. Esta capa sirve de enlace para que el sistema de razonamiento acuda a las capas

CIM OPERACIONAL Y DE DEFINICIÓN DEL DOMINIO, para realizar las inferencias necesarias durante el proceso de descubrimiento automático y de emparejamiento.

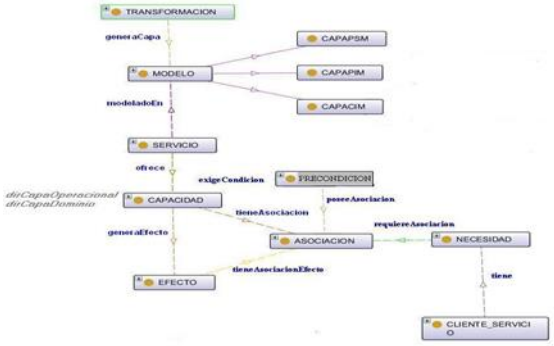


Figura 5. Modelo ontológico de la CAPA CIM REFERENCIAL

En la Figura 5 se muestra el modelo ontológico asociado a la subcapa CIMReferencial, en dicho modelo ontológico se describe que los servicios web descritos se ajustan en lo estructural a lo contemplado en la arquitectura SOA. Para ello, el modelo ontológico describe como un cliente tiene necesidades las cuales son emparejadas mediante métodos de emparejamiento con las capacidades ofrecidas por el proveedor de un servicio. Se describe además que tales capacidades generan efectos, los cuales consisten en asociaciones ya determinadas en la definición semántica del domino.

Lo que busca la capa es que durante el proceso de descubrimiento automático, composición automática y ejecución automática, el sistema de inferencia logre realizar un primer emparejamiento semántico considerando las precondiciones y efectos de cada capacidad y las necesidades solicitadas por el cliente. Es la capa CIMREFERENCIAL la capa de mayor jerarquía dentro de las tres subcapas componentes de la capa CIM, pues a partir de ella se acceden las otras capas representados en modelos ontológicos.

En la Tabla 5 se ilustra la especificación de capacidades, precondiciones y efectos definidos para el SWS del caso de prueba. También, se muestra la necesidades descritas semánticamente por la aplicación que invoca a WS GENERADOR CLIENTE. Como se aprecia en la Tabla 2, la acción entregar empareja con la acción recibir. Lo cual hace que el sistema de razonamiento infiera que la capacidad del SWS empareja con la necesidad del estudiante, pues lo que el estudiante necesita recibir (recomendacionOA), el SWS está en capacidad de entregar. Como se aprecia, las PRECONDICIONES son satisfechas, pues el recOA necesita como precondicion saber que tema debe enseñar, que es el mismo tema que el estudiante necesita aprender. Como el sistema de inferencias determina que enseñar y aprender emparejan (ver Tabla 5), el WS GENERADOR CLIENTE determina, a través de su módulo de razonamiento, que esta precondición es satisfecha.

	tieneAntecedente	tieneAccion	tieneConsecuente
CAPACIDAD	recOA	entregar	recomendacionOA
PRECONDICION	recOA	recibir	recomendacionOA
	recOA	enseñar	tema
EFECTO	estudiante	recibir	recomendacionOA
NECESIDAD	estudiante	recibir	recomendacionOA
	estudiante	aprender	tema

Tabla 5. Definición de capacidades, precondiciones y efectos en CAPA CIM REFERENCIAL

IV.A.1.b) DISEÑO E IMPLEMENTACIÓN DE LA CAPA CIM OPERACIONAL

En la CAPA CIMOPERACIONAL se especifica el modelado del negocio y sus requerimientos. Es la capa encargada describir, no solo las funcionalidades del proces, y quien lo usa, sino que además describe el proceso que se implementa y ejecuta en el servicio web, y la dinámica de interacción que surge en el momento de ejecución.

La Figura 6 muestra el modelo ontológico que soporta la capaCIMOperacional; el concepto central es CAPACIDAD, y este se relaciona con tres conceptos fundamentales, que son CASOUSO (funcionalidad), DIAGRAMAACTIVIDADES (flujo del proceso) y DIAGRAMADESECUENCIA (dinámica de interacción).

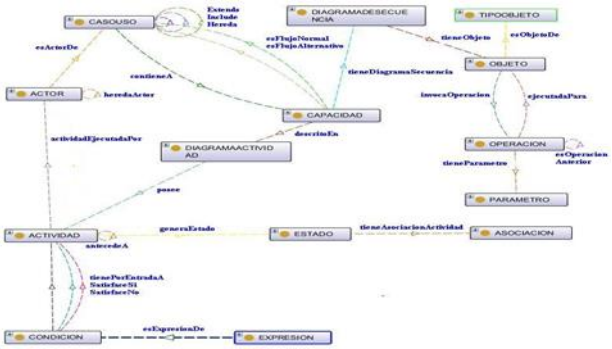


Figura 6. Modelo ontológico de la capa CIM OPERACIONAL

En torno al concepto CASODEUSO, se describen las distintas funcionalidades. El concepto DIAGRAMAACTIVIDAD es el encargado de organizar lo referente a la descripción del flujo de las actividades que componen el proceso, Mediante el concepto DIAGRAMADESECUENCIA se agrupa la descripción de los distintos objetos que interactúan a lo largo de la ejecución del proceso.

IV.A.2) DISEÑO E IMPLEMENTACIÓN DE LA CAPA PIM

En la Figura 7 se aprecia el modelo ontológico asociado a la capa PIM. La estructura fundamental del modelo ontológico de la capa PIM, está basada en lo propuesto en los diagramas de clases. Las reglas de transformación lo que buscan es que el razonador infiera el conocimiento necesario para generar el ABOX.

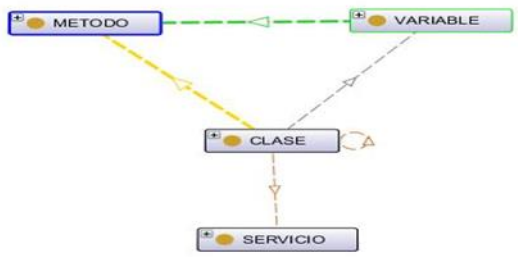


Figura 7. Modelo ontológico de la capa PIM.

El Framework FODAS-WS realiza un proceso de validación. Para ello, FODAS-WS cuenta con un analizador y validador, que se encargan de comprobar la congruencia entre la capa CIM OPERACIONAL, la capa CIMREFERENCIAL y LA CAPA DE DEFINICIÓN SEMÁNTICA DEL DOMINIO. El validador garantiza que cada uno de los aspectos especificados en la capa operacional y referencial, se encuentren descritos semánticamente en la definición semántica del dominio. El validador debe estar incorporado en la transformación CIMPIM, de manera que no se pueda generar la capa PIM en caso de que haya alguna inconsistencia.

IV.A.3) DESCRIPCIÓN DE LA GENERACIÓN AUTOMÁTICA DE LA CAPA PIM

La transformación CIMPIM de FODAS-WS realiza una primera generación de la capa PIM, en la que se cuenta con individuos instanciados como clases, como parámetros y como métodos. En el caso específico del individuo llamado RecOA, se determina que las reglas de transformación generaron que la RecOA tiene tres métodos llamados armar, ordenar y emparejarOA (todos vinculados en la capaPIM con el objectProperty tieneMetodo). También, es importante recalcar que en la transformación se genera para la clase RecOA, las variables llamadas razonador, arreglo, ontologia y estudiante (vinculadas en la capaPIM con el objectProperty tieneVariable). De igual forma, para el individuo llamado RecWeb, la transformación generó tres métodos, llamados armar, ordenar y emparejarWeb.

A partir de esta transformación básica, es necesario considerar la posible existencia de jerarquías de clases. La Figura 14 ilustra el resultado obtenido en la capa PIM, después de ejecutada esta regla de transformación en el caso de estudio. Como se aprecia en la Figura 14, la regla de transformación detectó que la RecOA y la RecWeb tienen métodos comunes (armar y ordenar) y variables comunes (ontología, arreglo y razonador), por ello genero una nueva clase, llamada RecOARecWeb, que contiene los métodos y variables comunes. Adicional a eso, indica que clases heredan de ella mediante el objectProperty esSuperClaseDe, con el cual se indica que las clases RecOA y RecWeb heredan de la super clase RecOARecWeb (nombre obtenido de concatenar los nombres de las clases que heredan de ella).

Además de generar la clase de nivel superior, las reglas de transformación también se encargan de eliminar de las subclases los métodos y variables, que se hereden de la clase padre. La figura 8 indica, por ejemplo, como en la clase RecOA la regla de transformación retiro, tanto los métodos llamados armar y ordenar, como las variables llamadas ontología, arreglo y razonador, pues ellas son heredadas de la RecOARecWeb.

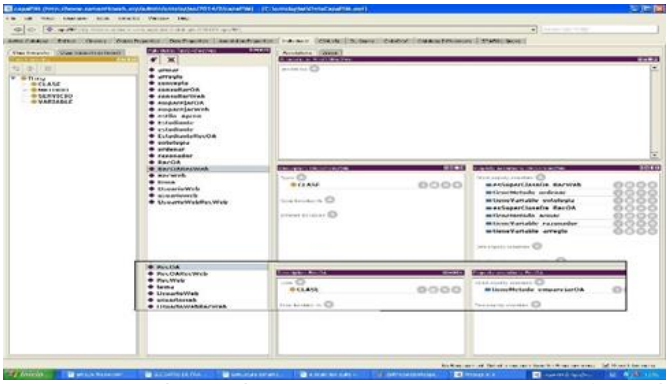


Figura 8. Generación automática de la capa PIM por medio de la transformación CIMPIM.

B. DISEÑO E IMPLEMENTACIÓN DE LA CAPA PSM

Esta capa especifica aspectos análogos a los especificados en el archivo WSDL asociado al Web Service, con la diferencia que en esta capa del FODAS-WS, el razonador puede realizar procedimientos formales de razonamiento, y además, contemplar que este solo es el nivel más alto de especificación del servicio; a diferencia de los servicios web tradicionales, en los que su descripción WSDL es la única proveída al cliente, y en un formato sin connotación semántica alguna.

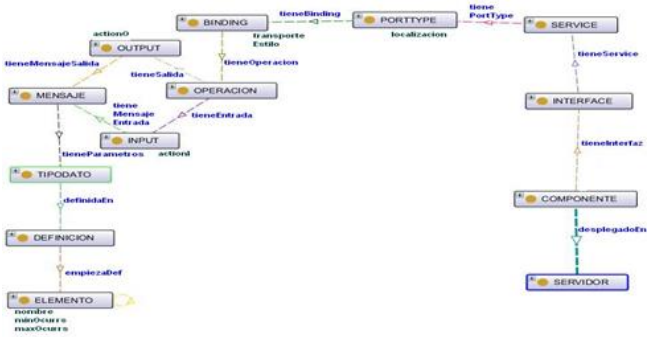


Figura 9. Modelo ontológico de la CAPA PSM

La Figura 9 muestra la capa PSM. Se observa que los componentes (servicios web) cuentan con una o varias interfaces, representadas mediante el concepto INTERFACE. Una vez WS GENERADOR CLIENTE tenga formalizado ésto, debe acudir a los demás niveles de abstracción, para entender el orden en que debe enviar los distintos mensajes, y lo que significa cada uno de los mensajes intercambiados con el servicio web.

V. ANÁLISIS DE RESULTADOS

La Tabla 6 muestra los resultados obtenidos al realizar una comparación de FODAS-WS con otros Framework existentes, que aunque no son totalmente idénticos en funcionalidad, pertenecen a la misma área de estudio [39], [40] y [41]. Los

resultados obtenidos con FODAS-WS se evalúan mediante una comparación de sus capacidades propuestas, respecto de las capacidades postuladas por esos trabajos similares. A continuación se enumeran los criterios de comparación entre los Framework usados:

1. Se ajusta a la arquitectura ODA
2. Tiene transformaciones automáticas.
3. Propone mecanismos de descubrimiento, composición y ejecución automáticos.
4. Poseé validador de semántica propio.
5. Propone ofrecer funcionalidades completas de composición mediante un servicio WS
6. Propone generar código esqueleto para la implementación del servicio
7. Propone generar código del cliente automático para implementar composición y ejecución automática.
8. Genera PSM a partir del servicio ya implementado, para garantizar total coherencia.
9. Tiene ontologías para representar cada capa
10. Usa estándares para representar las capas ontológicas

Criterio	FODAS-WS	SWF [14]	MDF-DWSOA [15]	METEOR-S [16]
1	SI	SI	NO	NO
2	SI	NO	NO	NO
3	SI	NO	NO	NO
4	SI	NO	NO	NO
5	SI	NO	NO	NO
6	SI	NO	SI	NO
7	SI	NO	NO	NO
8	SI	SI	NO	SI
9	SI	SI	NO	SI
10	NO	SI	SI	NO

Tabla 6. Comparativo entre FODAS-WS y otros trabajos

Es importante en este análisis considerar, que FODAS-WS muestra ser el framework más completo, pues además de permitir una descripción semántica completa del servicio, deja plasmado la conceptualización necesaria para realizar el descubrimiento, composición y ejecución automática de servicios web.

VI. CONCLUSIONES Y TRABAJOS FUTUROS

El uso de FODAS-WS propuesto cuenta con las siguientes ventajas:

- Usando los modelos ontológicos generados en el proceso de diseño, un razonador puede chequear los componentes de un sistema y verificar su consistencia.
- Con el uso de diseños ODA se pueden realizar inventarios de software, en los que además de conocer el software existente, se logre realizar un chequeo de funcionalidades repetidas o inconsistentes.
- El diseño de SWS mediante FODAS-WS representa un mecanismo de diseño comprensible para agentes de software, y el principal insumo para lograr descubrimiento, emparejamiento y composición automática de servicios web.
- El uso de FODAS-WS simplifica el proceso de diseño a través de la utilización de transformaciones automáticas entre capas, y la generación automáticas de las capas superiores.
- Un SWS diseñado con FODAS-WS no requiere de procesos de descripción semántica adicionales al diseño de la aplicación. Esto minimiza la posibilidad de que la aplicación no se ajuste a lo descrito en tales descripciones semánticas, y reduce los tiempos de diseño y especificación, ya que a medida que se está diseñando la aplicación también se está describiendo semánticamente, eliminando con ello un proceso adicional que puede dar pie a errores humanos en la descripción del servicio.
- Con un servicio web semántico diseñado en FODAS-WS, no solo se describe los aspectos básicos contemplados en otros procedimientos de especificación semántica (como OWL-S o WSMO), sino que se permite conocer la totalidad del diseño del servicio, no solo en lo funcional sino también en lo estructural.
- La realización de tareas mantenimiento de los servicios web se hace de manera más clara y congruente, ya que el razonador puede entregar al encargado de mantenimiento información inferida que puede ser útil.
- FODAS-WS resulta ser el framework que propone una funcionalidad más completa para el área de descripción de SWS. Además, propone mecanismos de descubrimiento, composición y ejecución automática de SW, que posibilitan la automatización de procesos de negocio.

A partir del presente trabajo, se propone realizar los siguientes trabajos futuros:

- Diseñar e implementar un módulo que se encargue de generar el código del servicio web diseñado en forma automática.
- Diseñar e implementar un servicio web que ofrezca los servicios de descubrimiento, composición y ejecución automática de SWS diseñados con FODAS-WS, cuyos clientes especifiquen semánticamente sus necesidades.
- Diseñar e implementar un prototipo que use el framework FODAS-WS, para la automatización de procesos de negocio.

REFERENCIAS

[1] P. Pan, C. Wang, G. Horng, and S. Cheng. The development of an Ontology-Based Adaptive Personalized Recommender System. in Electronics and Information Engineering (ICEIE), 2010 International Conference On. 2010.

[2] OBJECT MANAGEMENT GROUP, OMG (2003). MDA Guide Version 1.0.1 [online]. Massachusetts (USA): Object Management Group. http://www.omg.org/cgi-bin/doc?omg/03-06-01.pdf

[3] TIMM, Jhon T. E. & GANNOD, Gerald C. A model-driven approach for specifying semantic Web services. In: 3rd. IEEE International Conference

on Web Services, 2005, ICWS 2005, IEEE. Proceedings of the ICWS 2005, Vol. 1, p. 313-320. ISBN: 0-7695-2409-5.
[4] HYUK YANG, Jin & JEONG CHUNG. Automatic Generation of Service Ontology from UML Diagrams for Semantic Web Services. In: First Asian conference on The Semantic Web, ASWC 2006, Beijing (China). p. 523-529. ISBN: 3-540-38329-8 978-3-540-38329-1
[5] Pressman R. Ingeniería de Software. Un enfoque práctico. Ed. McGraw Hill Int. 2001.
[6] Bell, R. Code Generation from Object Models. 1998. Embedded Systems Programming. 74 – 88
[7] Herrington, J. Code Generation in Action. Greenwich: Manning Publications. 2003.
[8] UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado. Craig Larman. Prentice Hall. 2003
[9] Pinter, G., Majzik, I. Program Code Generation Based on UML Statechart Models. 2003. Periodica Polytechnica-Electrical Engineering. 187-204.
[10] Happel, H and Seedorf, S. Applications of Ontologies in Software Engineering. 2008.
[11] Anandaraj, A. ; Dheepak, G. and Raja, K. Study of Ontology in Software Modelling Process and Life Cycle. International Journal of Research and Reviews in Software Engineering Vol. 1, No. 1, March 2011 United Kingdom.[12] Montalvo, J. A multimedia ontology-driven architecture for autonomic quality of service management in home networks. Networking and Internet Architecture. INSA de Toulouse, 2012.
[12] P. Tetlow, J. Pan, D. Oberle, E. Wallace, M. Uschold, and E. Kendall. Ontology Driven Architectures and Potential Uses of the Semantic Web in Systems and Software Engineering. W3C Working Draft, 2006.
[13] Pahl, C. Layered Ontological Modelling for Web Service-oriented Model-Driven Architecture. Dublin City University. 2008.
[14] Achilleos, A ; Kapitsaki, G and Papadopoulos, G. A Model-Driven Framework for Developing Web Service Oriented Applications. Department of Computer Science, University of Cyprus. 2006.
[15] Patil, A; Oundhakar, S ;Sheth, A and Verma, K. METEOR-S: Semantic Web Services and Processes. 2005.
[16] Vega, W. & Umaña H. Diseño de Servicios Web Semánticos utilizando el desarrollo de software dirigido por modelos. En: Ventana Informática No. 30 (ene-jun). Manizales (Colombia): Facultad de Ciencias e Ingeniería, Universidad de Manizales. p. 97-108. ISSN: 0123-9678. 2014.
[17] KALANTARI, Alaeddin; IBRAHIM, Suhaimi & TAHERDOOST, Hamed. A categorization of model-driven approaches for developing Semantic Web Service. ISBN: 978-94-007-2791-5. 2011.
[18] KIM, Il-Woong & LEE, Kyong-Ho. A Model-Driven Approach for Describing Semantic Web Services: From UML to OWL-S. ISSN: 1094-6977. 2009.
[19] TIMM, Jhon T. E. & GANNOD, Gerald C. A model-driven approach for specifying semantic Web services. In: 3rd. IEEE International Conference on Web Services, 2005, ICWS 2005, IEEE. Proceedings of the ICWS 2005, Vol. 1, p. 313-320. ISBN: 0-7695-2409-5.
[20] Burke, R., Hybrid recommender systems: Survey and experiments. User Modeling and User-Adapted Interaction, 2002. 12(4): p. 331-370.
[21] Bobadilla, J. Recommender systems survey, 2013. España

Jose Aguilar is a System Engineer graduated in 1987 from the Universidad de los Andes, Merida, Venezuela. M. Sc. degree in Computer Sciences in 1991 from the University Paul Sabatier-Toulouse-France. Ph. D degree in Computer Sciences in 1995 from the University Rene Descartes-Paris-France. He completed post-doctorate studies at the Department of Computer Science in the University of Houston, United States, between 1999 and 2000. Titular Professor at the Department of Computer Science in the Universidad de los Andes and researcher at the Microcomputer and Distributed Systems Center (CEMISID) at the same university. Member of the Mérida Science Academy and the International Technical Committee of the IEEE-CIS on Artificial Neural Networks.

Omar Portilla es Ingeniero de Sistemas graduado en 2001 de la Universidad Industrial de Santander, Bucaramanga, Colombia. Candidato a Magister en Computación de la Universidad de los Andes Mérida. Profesor del programa de Ingeniería de Sistemas de la Universidad de Pamplona, Colombia.

Middleware Reflexivo para la gestión de Aprendizajes Conectivistas en Ecologías de Conocimientos (eco-conectivismo)
Reflective Middleware for Managing Learning Connectivism in Knowledge Ecologies (eco-Connectivism)

Jose Aguilar y Diego Mosquera

***Resumen*— En este artículo se propone la arquitectura de un Middleware Reflexivo basado en computación autonómica, cuyo objetivo es gestionar un ambiente conectivista de aprendizaje, modelado bajo el paradigma de las ecologías del conocimiento. El Middleware es capaz de monitorear el ambiente que consiste de un conjunto de Entornos Personales de Aprendizaje que son percibidos como objetos auto-organizados que forman ecosistemas. La evolución del proceso de aprendizaje depende del análisis del comportamiento Web de los aprendices, y de un esquema de supervivencia ecológica que promueve las relaciones sociales, diversidad y tolerancia en un dominio de conocimiento socializado. El middleware utiliza minería web de uso para caracterizar el comportamiento del aprendiz, técnicas de agrupamiento para los ecosistemas de aprendizaje, y un sistema recomendador cognitivo-colaborativo para el proceso de auto-adaptación de las estrategias de aprendizaje.**

***Palabras clave*— Clustering (agrupamiento), Conectivismo, Ecología de Conocimiento, Entorno Personal de Aprendizaje, Middleware Reflexivo, Minería Web, Sistemas de Recomendación.**

***Abstract*— This article describes the architecture of a reflective middleware based on autonomic computing, with the goal of managing a connectionist learning environment, modeled following the paradigm of knowledge ecologies. The middleware is able to monitor the environment consisting of a set of personal learning environments that are perceived as self-organized objects forming ecosystems. The evolution of the learning process depends on the analysis of web behavior of students, and the ecological survival scheme that promotes social relations, diversity and tolerance in socialized domain knowledge. The middleware uses web mining to characterize the behavior of the student, clustering techniques for the learning ecosystems, and a cognitive-collaborative recommender system for self-adaptation process of learning strategies.**

***Index Terms*— Clustering, Connectivism, Ecology of Knowledge, Personal Learning Environment, Reflective Middleware, Web Mining, Recommendation Systems.**

I. INTRODUCTION

LAS Tecnologías de Información y Comunicación (TICs), en especial, las relacionadas con la Web social y la Web inteligente, han traído consigo una revolución en la educación, en cuanto han transformado el concepto de aprendizaje, para adaptarlo a nuevas estructuras cognitivas tanto individuales como colectivas. Por otro lado, el cúmulo de Información que se precisa en Internet, las comunidades especializadas, los repositorios de contenidos, los servicios para publicación de conocimiento especializado, y las múltiples herramientas de cooperación, han puesto de manifiesto nuevos esquemas de aprendizaje, donde las relaciones sociales, la interacción y la auto-organización son las principales propiedades del proceso. A estos nuevos espacios, y por sus propiedades emergentes, se les denomina ecología del conocimiento.

En este sentido, han surgido diversas propuestas en las que se mezclan planteamientos conocidos, como el constructivismo y la teoría de la complejidad, para capturar el efecto que las TICs han tenido sobre el aprendizaje, y con ello intentar formular descripciones precisas de los procesos de aprendizaje cuando éstos son mediados por las tecnologías. Ejemplos de estas propuestas son: aprendizaje rizomático, la heutagogía, la paragogía, la pedagogía de la proximidad y el conectivismo. No obstante, al intentar caracterizar medios computacionales de

Dr. Aguilar ha sido parcialmente financiado por el Proyecto Prometeo del Ministerio de Educación Superior, Ciencia, Tecnología e Innovación de la República de Ecuador).

J.A. Universidad de Los Andes, Facultad de Ingeniería, Escuela de Ingeniería de Sistemas, CEMISID, Mérida, E-mail: aguilar@ula.ve. Además, J. A. actualmente es Investigador Prometeo en la Universidad Técnica Particular de Loja, Loja, Ecuador.

Diego Mosquera. is professor at the Department of Science and Tecnology in the Universidad Nacional Experimental de Guayana, UNEG, Venezuela.

gestión para este tipo de entornos, surgen serios problemas de completitud en las propuestas. Por ejemplo, tanto la heutagogía como la paragogía limita el contexto de aprendizaje a clases de individuos; el aprendizaje rizomático y el conectivismo no definen métodos ni modelos para medir y validar el aprendizaje (calidad de las conexiones); el conectivismo se propone como una teoría de aprendizaje, cuando realmente carece de varios elementos propios del diseño de teorías.

Sin embargo, al combinar el concepto de ecología de conocimiento con el fundamento teórico del conectivismo, es posible encontrar un acoplamiento entre el modelo abstracto descrito por las teorías de aprendizaje, y un modelo computacional que gestione el proceso incluyendo, mecanismos de validación.

El paradigma de ecologías del conocimiento se basa en el estudio y análisis de las interacciones sociales que se producen en un entorno de aprendizaje emergente en relación con la información, las tecnologías, la generación del conocimiento y el ambiente que lo rodea [1]. Estos entornos de aprendizaje pueden ser explicados en el marco epistemológico del conectivismo, y guiado pedagógicamente por una dinámica evolutiva estrechamente relacionada con el concepto de ecología [1].

Asimismo, las ciencias computacionales ofrecen diversas metodologías y herramientas que pueden ser utilizadas para caracterizar el proceso de aprendizaje conectivista que se produce en una ecología de conocimiento [2]. Por ejemplo, la minería web de uso permite el descubrimiento automático de patrones de comportamiento o uso de servicios de los internautas en la Web [3]. Igualmente, los algoritmos de agrupamiento facilitan asociar elementos de acuerdo a criterios de distancia o similitud [2, 3]. Finalmente, los sistemas de recomendación permiten el filtrado de información basado en criterios de acercamiento [4].

El objetivo de este artículo es proponer un esquema de integración de un modelo de aprendizaje conectivista, con técnicas propias de las ciencias de la computación (por ejemplo, minería semántica y de datos), para modelar la dinámica que se produce en una ecología de conocimiento integrada por Entornos Personales de Aprendizaje (o PLEs, por sus siglas en inglés) [5].

En particular, la minería web de uso es utilizada para monitorear y analizar la actividad de los aprendices, de tal forma que se pueda conocer su patrón de comportamiento. El resultado de este análisis permitirá caracterizar (tipificar) al individuo conforme a ese patrón de aprendizaje. El agrupamiento permite asociar/agrupar a los aprendices de acuerdo a los resultados obtenidos en el proceso de tipificación apoyada en la dinámica pedagógica propia de la teoría de aprendizaje conectivista. Con esta información es posible definir ecosistemas de PLEs y así planificar rutas de aprendizaje, en función a las necesidades pedagógicas particulares de cada grupo. Durante el proceso de aprendizaje, se calcula la tasa de renovación ecológica para cada ecosistema, la cual es función de las conexiones entre PLEs que se generan como parte del comportamiento social y colaborativo de los aprendices. Aquellos ecosistemas con tasas de renovación bajas (de acuerdo a un umbral predefinido) desaparecen de la ecología. Cuando un ecosistema se elimina, el conjunto de PLEs involucrados deben reinsertarse en aquellos ecosistemas que agrupen los PLEs con caracterísitcas similares. Luego un sistema recomendador híbrido (cognitivo-colaborativo) interpreta cómo ciertas características de determinados recursos digitales cumplen con las necesidades de reinserción del aprendiz, basándose en la utilidad que estos recursos han tenido sobre los aprendices incluidos en el ecosistema receptor.

La estructura del artículo es la siguiente: La sección 2 presenta el marco teórico con material referencial relacionado con los paradigmas y teorías de aprendizaje en uso; las tecnologías computacionales de la Web y los middlewares reflexivos autonómicos. La sección 3 describe el método utilizado para la especificación del middleware de acuerdo al alcance actual de la investigación propuesta en este artículo. La sección 4 presenta el diseño funcional de la arquitectura. Finalmente, en la sección 5 se precisan las conclusiones y los trabajos futuros.

II. MARCO TEÓRICO

A. Ecologías de conocimiento, aprendizaje conectivista y Entornos Personales de Aprendizaje

El paradigma de ecologías del conocimiento se basa en el estudio y análisis de las interacciones sociales que se producen en un entorno de aprendizaje emergente en relación con la información, las tecnologías, la generación del conocimiento y el ambiente que lo rodea [1]. Una ecología del conocimiento es un conjunto de Redes Personales de Aprendizaje (PLNs, por sus siglas en inglés) interrelacionadas, que forman un sistema complejo con entidades auto-organizadas y propiedades emergentes [5]. Una PLN conforma el hogar del conocimiento y la identidad del aprendiz individual, representado por un repertorio adaptativo de herramientas sociales, procesos mentales y actividades para compartir, reflexionar, discutir y reconstruir con otros el conocimiento; además de las actitudes que propician y nutren ese intercambio [6]. En una ecología del conocimiento, el aprendizaje es consecuencia de la extensión de las PLNs con nuevos nodos de conocimiento. Es importante destacar las PLNs forman parte de una estructura mayor denominada Entorno Personal de Aprendizaje (PLE, por sus siglas en inglés) [5, 6].

Un PLE se define como el conjunto de herramientas, fuentes de información, conexiones y actividades que cada individuo utiliza de forma asidua para aprender. Un PLE tiene tres componentes: 1) fuentes documentales y experienciales; 2) medios de reflexión; y 3) medios de interacción. Cada componente del PLE tiene asociado un conjunto de herramientas, mecanismos de aplicación y actividades de aprendizaje [5].

El conectivismo es una teoría de aprendizaje emergente que busca dar respuesta a la influencia que ha tenido la tecnología en aprendizaje [7]. El conectivismo pretende explorar y explotar los procesos metacognitivos del ser humano, combinando los objetivos tradicionales del aprendizaje “saber ser” (actitud), “saber cómo hacer” (métodos) y “saber qué hacer” (contenidos), con estrategias de autoregulación como “saber dónde buscar” y “saber transformar” acorde a estilos de aprendizaje [5, 7]. Para el conectivismo, lo importante ya no es la adquisición de gran cantidad de conocimientos, sino la

capacidad de adaptación a un mundo en constante cambio y las estrategias metacognitivas para transformar el conocimiento. Esto último basado en la premisa de que las conexiones proporcionan mejores resultados que el intento de comprender los conceptos de manera individual [7]. Para el conectivismo, el aprendizaje es visto como un proceso emergente de construcción de redes especializadas en un entorno diverso y complejo [1]. Epistemológicamente, el conectivismo tiene sus bases en el conocimiento socialmente distribuido y la actividad situada [8], basado en la percepción de que nadie posee todo el conocimiento necesario y que, por el contrario, éste se encuentra distribuido entre aquellos que poseen objetivos comunes; el aprendizaje es logrado cuando se establecen conexiones entre entidades especializadas. Un aspecto importante que introduce el conectivismo es el de patrones en aprendizaje como la base de la formación [7, 1]. En términos específicos, una teoría conectivista debe incorporar las estrategias necesarias para generar, en los individuos, habilidades de reconocer conexiones entre áreas, ideas y conceptos; al mismo tiempo de proporcionar las estrategias de transformación del conocimiento para que éste se mantenga preciso y actualizado [1].

B. Tecnologías Computacionales de la Web

En este trabajo se propone el uso de varias técnicas del área de las tecnologías Web para el desarrollo de la plataforma. En particular es de interés la minería web de uso, las técnicas de minería de datos de agrupamiento y los sistemas recomendadores.

- La *minería web* de uso es un tipo de minería Web que utiliza algoritmos de minería de datos y semántica con el objetivo de descubrir patrones conductistas de la actividad y comportamiento de los usuarios, relacionados con la navegación Web en Internet [3]. Tal como se puede observar en la Figura 1, la minería Web de uso abarca cuatro etapas:

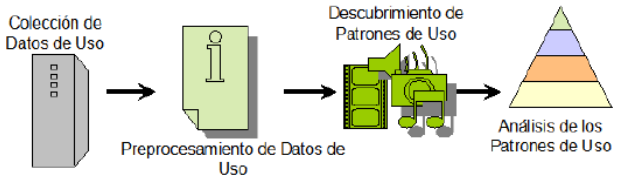


Figura 1: Etapas de la Minería Web de uso

La etapa de colección de datos de uso está relacionada con los datos que pueden ser extraídos de los registros (logs) de los dispositivos físicos: servidores Web, servidores proxy y máquinas de usuarios; además de la información que se obtiene de las estructuras que forman los enlaces hipertexto en las páginas Web. El preprocesamiento de datos de uso lleva a cabo el procesamiento inicial de los archivos logs capturados en la fase anterior; y el objetivo principal es el filtrado de la información y la generación de unidades lógicas por sesiones de usuario. En la etapa de descubrimiento de datos de uso se caracterizan los patrones; en esta fase normalmente se utilizan las técnicas de análisis en la minería de datos. Finalmente, en la etapa de análisis de patrones de uso se filtran las reglas y patrones de interés.

- El *agrupamiento* consiste en definir grupos de objetos de acuerdo a las propiedades comunes que comparten [2]. El conocimiento de los grupos permite una descripción sintética del conjunto basados en datos multidimensionales complejos, que se consiguen sustituyendo la descripción de todos los elementos de un grupo por un representante característico del mismo (a veces llamado centroide) [2, 3]. Existen diversas técnicas para aplicar un procedimiento de clustering; algunas basadas en agrupamiento jerárquico como NTREE, y otras en agrupamiento no-jerárquico como K-MEANS, centroides SOM o redes de Kohonen [3].
- Los *sistemas recomendadores* son un tipo específico de técnica de filtrado de información Web que se basan en emparejar distintos tipos de temas/ítems/etc. de acuerdo a los intereses de un usuario particular. Generalmente un sistema recomendador compara el perfil del usuario con características de referencias de los temas, y predice la ponderación que un usuario le daría a un ítem que aún no ha sido considerado. Los sistemas recomendadores tienen diversas aplicaciones en Internet: 1) en e-comercio para ofrecer servicios personalizados al cliente; 2) en los buscadores de páginas web para filtrar la información; 3) en las bibliotecas digitales para ayudar a encontrar los libros o artículos que se ajustan a las preferencias del usuario; entre otros. Existen diversas estrategias para caracterizar sistemas de recomendación, los cuales dependen de los objetivos de la aplicación: 1) recomendación colaborativa, basada en el principio de popularidad entre los usuarios; 2) recomendación de contenido, que usa el principio de las preferencias del usuario; 3) recomendación en conocimiento, basado en el principio de las necesidades del usuario; y 4) híbridas, combinaciones de dos o más estrategias en un mismo sistema [4].

C. Middlewares Reflexivos y Computación Autonómica

Un middleware reflexivo es un sistema que actúa como capa intermedia entre aplicaciones y servicios, que tiene la habilidad de observar y cambiar su propio comportamiento a través de un proceso de auto-referencia y auto-conciencia. En general, la parte reflexiva de un middleware se implementa acoplando dos procesos [9]:

- **Introspección**: Habilidad para observar y sobre su propio estado de ejecución.
- **Intercepción**: Habilidad para modificar su propio estado de ejecución, o alterar su propia interpretación (o significado).

Un middleware reflexivo consiste de dos o más niveles de agregación:

- **Nivel base**: Donde se ejecutan las aplicaciones y funcionalidades propias del sistema.

- **Nivel meta**: Implementa la reflexividad y verifica que las operaciones del sistemas sean las requeridas o esperadas.

La computación autonómica es un modelo de auto-gestión inspirado en el sistema nervioso de los seres humanos. Este incorpora sensores y actuadores para observar el ambiente, razonar y actuar en consecuencia. Para modelar las propiedades autonómicas de un sistema se introduce un modelo de referencia que integra los siguientes elementos [10]:

- **Elementos manejados**: Cualquier recurso (hardware o software) embebido en el sistema que tiene atributos que pueden ser auto-gestionados.
- **Sensores**: Colección de mecanismos de captación de información sobre los elementos manejados.
- **Manejador autonómico**: Implementa los lazos de control inteligentes que automatizan las tareas de autorregulación de las aplicaciones. Está compuesto por cuatro módulos que caracterizan el lazo de control autónomo: Monitoreo, Análisis, Planificación y Ejecución (MAPE). El módulo de monitoreo recolecta los eventos/datos de los sensores. El módulo de análisis identifica y examina las situaciones de interés. El módulo de planificación decide y organiza las tareas a realizar a partir de estados particulares del sistema y el conocimiento interno representado. El módulo de ejecución permite enviar los resultados obtenidos a los actuadores.
- **Actuadores**: Lleva a cabo los cambios en los elementos manejados.

III. ALCANCE Y MÉTODO

El alcance de este artículo es la especificación de una arquitectura de un Middleware Reflexivo basado en computación autonómica, cuyo objetivo es gestionar un ambiente conectivista de aprendizaje, modelado bajo el paradigma de las ecologías del conocimiento. Esto se logra cumpliendo tres fases:

1. Definición de las bases de un proceso de aprendizaje eco-conectivista.
2. Diseño de la arquitectura reflexiva autonómica.
3. Especificación de los diferentes componentes de la arquitectura.

En este sentido se utilizan dos marcos de referencia: 1) la reflexión computacional y sus niveles de introspección e intersección y 2) el modelo de referencia MAPE para la especificación de las propiedades autonómicas del sistema.

Para caracterizar la dinámica de aprendizaje en cuanto a los momentos pedagógicos, se define un entorno de aprendizaje inspirado en las etapas de la sucesión ecológica y la biología de los ecosistemas. A este tipo de entorno lo denominamos eco-conectivismo.

IV. ARMAGAECO-C

Definimos el eco-conectivismo como un ambiente de aprendizaje conectivista sensible al contexto, gestionado por tecnologías computacionales, cuyo modelo pedagógico está inspirado en el concepto de sucesión ecológica. El eco-conectivismo es una extensión del conectivismo que define, de manera precisa, los momentos pedagógicos del aprendizaje como un proceso evolutivo y auto-adaptativo de PLEs usando conceptos provenientes de la ecología.

La ecología se define como la especialidad científica centrada en el estudio y análisis del vínculo entre los seres vivos y el entorno que los rodea, entendido como la combinación de los factores abióticos (como el clima y la geología) y factores bióticos (organismos que comparten el hábitat). La ecología también analiza la distribución y cantidad de organismos vivos como resultado de las mencionadas relaciones. En una ecología existen diversos conceptos que permiten la dinámica evolutiva; entre los cuales están:

- **Biocenosis:** conjunto de organismos de todas las especies que coexisten en un espacio definido llamado hábitat, que ofrece las condiciones ambientales necesarias para su supervivencia.
- **Ecosistema**: Es un sistema natural que está formado por un conjunto de organismos vivos y el medio físico donde se relacionan.
- **Hábitat:** Área de condiciones ambientales uniformes que provee espacio vital para que la especie pueda residir y reproducirse, perpetuando su presencia.
- **Biomasa**: Cantidad de materia acumulada en un individuo, población o ecosistema.
- **Tasa de renovación**: Es la relación que existe entre la producción y la biomasa. Sirve para indicar la riqueza de un ecosistema. Su valor es el cociente entre la producción neta y la biomasa.
- **Biodiversidad:** Se refiere a la amplia variedad de seres vivos sobre la Tierra y los patrones naturales que la conforman, resultado de años de evolución según procesos naturales.

El eco-conectivismo toma estos conceptos propios de la ecología y el de sucesión ecológica para caracterizar un sistema de gestión de servicios a procesos de aprendizaje conectivista llamado ARMAGAeco-c.

En el eco-conectivismo cada aprendiz real se asocia con un componente lógico que abstrae su PLE. Este conjunto de objetos define la biocenosis del modelo, cuyo hábitat está caracterizado por los objetivos y competencias de aprendizaje; que permanecen invariantes a lo largo de todo el proceso. Una vez conformado este conjunto de actores, entran en juego las características y condiciones comunes de supervivencia, las cuales se relacionan con las capacidades sociales de cada aprendiz. Esto permite la conformación de ecosistemas con habitas particulares, cuyo agrupamiento depende tanto de un perfil de aprendizaje precalculado, como del número de interacciones entre los aprendices. Al tratarse de un proceso de

aprendizaje conectivista, estos dos factores de agrupamiento definen la biomasa del modelo.

Con esta asociación de conceptos definimos ARMAGAeco-c (Arquitectura Reflexiva con Multinivel de Autonomía para Gestión de Aprendizaje eco-Conectivista), un middleware reflexivo de comportamiento dinámico y auto-adaptativo, que permite caracterizar medios de gestión de servicios de aprendizaje en entornos eco-conectivistas. Los momentos pedagógicos de inicio y desarrollo son estados transitorios de aprendizaje que forman parte de un proceso cíclico que agrupa a los aprendices en clusters. Cada cluster es un conjunto de enlaces fuertes entre individuos (grupos con PLEs comunes y relacionados) que forma parte de la ecología. La ecología de conocimiento es representada por un conjunto de enlaces débiles entre clusters. Para lograr el aprendizaje conectivista, la dinámica evolutiva de la ecología debe responder a un proceso de reducción de clusters (buscando un entorno común cada vez más diverso y tolerante). La fase pedagógica de cierre se alcanza cuando es posible minimizar el número de clusters (idealmente uno) como función de los objetivos de aprendizaje y nivel de especialización.

ARMAGAeco-c se basa en la arquitectura multinivel clásica de los middlewares reflexivos, con tres niveles de agregación: un nivel base, donde se ejecutan las funcionalidades propias del sistema, y dos niveles meta para implementar las capacidades autonómicas [13, 14]. La Figura 2 muestra la arquitectura de ARMAGAeco-c y las relaciones MAPE.

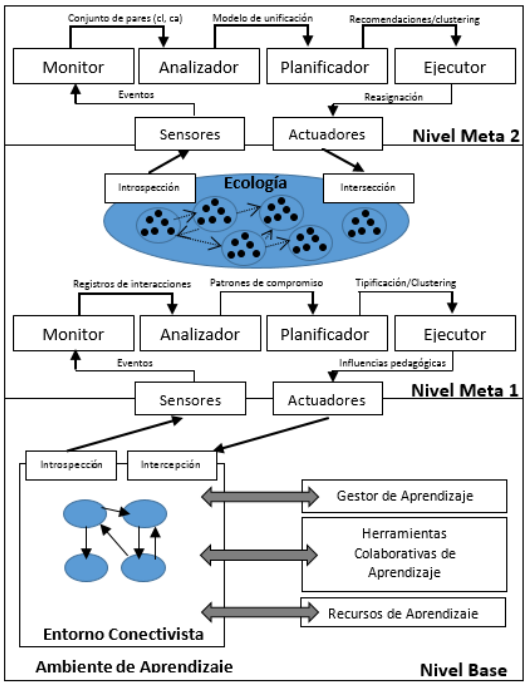


Figura 2: Arquitectura del Middleware

En el *nivel base* se encuentra el ambiente de aprendizaje (físico o virtual). Para cada aprendiz se instancia un componente lógico que representa su PLE. Inicialmente, los PLEs se construyen de acuerdo a un repertorio de reglas de asignación presentes en la base de conocimientos del Gestor de Aprendizaje (GA). Cada regla en el GA tiene la forma A→B, donde A se refiere al estilo de aprendizaje del estudiante, que es calculado por un servicio Web del sistema, y B es una conjunción que relaciona los ítems del PLE con recursos de aprendizaje.

La introspección del nivel base se hace analizando los atributos de los componentes lógicos en ejecución y del propio ambiente de aprendizaje (relaciones, objetivos, marco curricular, nivel de especialización, etc), los cuales definen una estructura abstracta (ecología), cuyos patrones comunes entre PLEs son agrupados a través de un proceso de agrupamiento. Para caracterizar el uso de esa estructura abstracta se utiliza un proceso de minería Web de uso que permite el descubrimiento de patrones conductistas de actividades y comportamiento de los aprendices. Toda esa información es usada por la técnica de agrupamiento para crear los clusters. Luego, el conjunto de ecosistemas (clusters) son interpretados por el middleware como una unidad lógica y distribuida denominada ecología del conocimiento.

El *nivel meta 1* del middleware tiene como objetivo caracterizar los nodos que definen los individuos y sus PLEs, para lo cual debe conocer las reglas evolutivas que definen y norman la supervivencia ecológica. La tabla 1 muestra los elementos MAPE en este nivel de reflexión:

Tabla 1: Elementos MAPE del primer nivel de reflexión

Componente MAPE	Rol en el sistema
Monitor	Inspecciona el entorno de aprendizaje y los eventos generados por los componentes lógicos presentes en el nivel base. El monitor implementa la primera y segunda etapa de un proceso de minería Web de uso, y emite señales de alerta al analizador para iniciar el proceso de descubrimiento de patrones.
Analizador	Analiza el comportamiento social del proceso de aprendizaje. Implementa la tercera etapa del proceso de minería Web de uso. El objetivo es llevar a cabo el análisis matemático de los datos para deducir patrones y tendencias de aprendizaje.
Planificador	Implementa la última etapa del proceso de minería Web de uso. Es el responsable de la tipificación de los aprendices de acuerdo a su patrón de comportamiento, y de establecer las reglas de asignación de recursos en el PLE de acuerdo a la evolución del aprendizaje. Para ambos procesos, el planificador dispone de un repertorio de reglas del tipo condición-acción, conocidas en MAPE como base de conocimiento.
Ejecutor	Es el responsable de llevar a cabo el plan pedagógico conectivista de acuerdo a los resultados obtenidos por el planificador. El ejecutor utiliza la base de conocimientos del Gestor de Aprendizaje para la asignación de recursos y, dependiendo del momento pedagógico del proceso de aprendizaje, utiliza el servicio Web para recalcular los estilos de aprendizaje.

La introspección del nivel meta 1 se hace analizando la ecología del conocimiento como una entidad completa.

Basado en ello el *nivel meta 2* hace una reflexión sobre la ecología de conocimiento que emerge del nivel meta 1, estudiando los clusters, sus grados de diversidad, etc; para lo cual utiliza la técnica de minería de datos de agrupamiento. En menos palabras, analiza el ambiente conectivista de aprendizaje y los niveles de especialización requeridos en el proceso de

aprendizaje que se está llevando a cabo (competencias a lograr, adquisición de conocimiento alcanzado, etc.). De esta manera, para cada cluster se calcula el potencial de supervivencia ecológica P_ve, cuyo valor es el cociente entre el número de enlaces débiles del cluster y el número de enlaces débiles en la ecología.

Sobre aquellos clusters que tienen la menor posibilidad de supervivencia, se activa un sistema recomendador híbrido (colaborativo y de conocimiento) que ayuda en la reubicación de los aprendices involucrados en el cluster más cercano, al mismo tiempo de ajustar el conjunto de recursos de aprendizaje. Para ello el sistema recomendador interpreta cómo ciertas características de determinados recursos digitales cumplen con las necesidades de reinserción del aprendiz, basándose en la utilidad que estos recursos han tenido sobre los aprendices incluidos en el ecosistema receptor. Ambos ajustes se deben relacionar con los PLEs. La tabla 2 muestra los elementos MAPE de este segundo nivel de reflexión de la arquitectura:

Tabla 2: Elementos MAPE del segundo nivel de reflexión

Componente MAPE	Rol en el sistema
Monitor	Hace seguimiento de los cambios producidos en la ecología de conocimiento en cuanto a los enlaces conectivistas débiles, a los PLE de los individuos, etc. Durante los momentos de aprendizaje, el monitor recolecta información sobre los ecosistemas (clusters) presentes en la ecología (nivel meta 1). Por cada ecosistema calcula el potencial de supervivencia ecológica P_ve y coloca una etiqueta de extinción a los ecosistemas con menores posibilidades de supervivencia. Luego, el monitor emite la señal de alerta correspondiente al analizador.
Analizador	Lleva a cabo una revisión de los PLEs involucrados en los ecosistemas etiquetados como "de extinción". Después se encarga de calcular las medidas de similitud de cada PLE con los clusters (ecosistemas) no-etiquetados. Las medidas de similitud se calculan usando el "criterio del testigo más fuerte" (PLE de los ecosistemas no-etiquetados con mayor cantidad de conexiones fuertes). Por cada aprendiz, se calculan k medidas de similitud (donde k es el número de ecosistemas no-etiquetados de la ecología).
Planificador	Es el responsable de llevar a cabo la incorporación del aprendiz en un ecosistema no-etiquetado, de acuerdo a los cálculos de medidas de similitud. Es decir, se da un poco de agrupamiento. Además, el planificador utiliza su base de conocimientos para rehacer el PLE del aprendiz que viene de migrar, recomendando los nuevos recursos de aprendizaje que debe incorporar cada aprendiz a su PLE para poder ser incorporado al ecosistema más similar (al que viene de ser asignado).
Ejecutor	Es el responsable de ejecutar el plan de actualización: eliminación de clustes etiquetados, reasignación de individuos y actualización de los PLEs de los individuos involucrados.

V. CONCLUSIONES Y TRABAJOS FUTUROS

En este artículo se ha presentado la propuesta inicial de una arquitectura de un middleware reflexivo y autonómico llamada ARMAGAeco-c, que puede ser implementada para caracterizar medios de gestión de servicios de aprendizaje en Internet. Este modelo explota un paradigma pedagógico que hemos llamado eco-conectivismo. Para ello, permite la creación de dinámicas de aprendizaje basadas en los principios teóricos del conectivismo para propiciar ambientes auto-organizados y auto-regulados de aprendizaje socializado sobre un dominio de conocimiento particular.

De esta manera, el concepto de eco-conectivismo, permite modelar el proceso de aprendizaje conectivista como una ecología "real", donde se integran elementos como ecosistemas, sucesión ecológica, biomasa, tasa de renovación, entre otras. En ese sentido, ARMAGAeco-c posibilita ese ambiente eco-conectivista cuyos organismos presentes en la ecología son Entornos Personales de Aprendizaje que se adaptan a un sistema cada vez más diverso, cuya estabilidad depende del nivel de especialización deseado en un momento dado. De esta manera, de forma dinámica, explotando sus propiedades de auto-regulacion y auto-organización, el sistema permite la emergencia de una ecología de aprendizaje adecuada a una temática deseada.

Finalmente, es importante destacar que en este artículo se presenta el diseño de ARMAGAeco-c, cuya fase de implementación y prueba se encuentra actualmente en proceso de desarrollo. En este sentido, se propone como trabajo futuro la publicación de las tecnologías computaciones utilizadas para su implementación, así como las pruebas y resultados de un prototipo experimental.

REFERENCES

[1] G. Siemens, Knowing knowledge, Creative Commons, 2006.

[2] A. Salem, «Intelligent Methodologies and Technologies for e-Learning,» *ICETA - 10th IEEE International Conference on Emerging eLearning Technologies and Applications,* pp. 331-337, 2012.

Jose Aguilar is a System Engineer graduated in 1987 from the Universidad de los Andes, Merida, Venezuela. M. Sc. degree in Computer Sciences in 1991 from the University Paul Sabatier-Toulouse-France. Ph. D degree in Computer Sciences in 1995 from the University Rene Descartes-Paris-France. He completed post-doctorate studies at the Department of Computer Science in the University of Houston, United States, between 1999 and 2000. Titular Professor at the Department of Computer Science in the Universidad de los Andes and researcher at the Microcomputer and Distributed Systems Center (CEMISID) at the same university. Member of the Mérida Science Academy and the International Technical Committee of the IEEE-CIS on Artificial Neural Networks.

Diego Mosquera is a System Engineer graduated in 2000 from de IUPSM, Venezuela. M.Sc. degree in Model and Simulation Systems from the Universidad de los Andes in 2005. Doctoral Student of Engineering Sciences from UNEXPO, Venezuela. Instructor Professor at the Department of Science and Tecnology in the Universidad de Guayana - UNEG and researcher at the Emergent Computation Center at the same university.

Human Activity Recognition in a Car with Embedded Devices

Danilo Burbano and Jose Luis Carrera

***Abstract*—Detection and prediction of drowsiness is key for the implementation of intelligent vehicles aimed to prevent highway crashes. There are several approaches for such solution. In this paper the computer vision approach will be analysed, where embedded devices (e.g. video cameras) are used along with machine learning and pattern recognition techniques for implementing suitable solutions for detecting driver fatigue.**
Most of the research in computer vision systems focused on the analysis of blinks, this is a notable solution when it is combined with additional patterns like yawing or head motion for the recognition of drowsiness. The first step in this approach is the face recognition, where AdaBoost algorithm shows accurate results for the feature extraction, whereas regarding the detection of drowsiness the data-driven classifiers such as Support Vector Machine(SVM) yields remarkable results.
One underlying component for implementing a computer vision technology for detection of drowsiness is a database of spontaneous images from the Facial Action Coding System (FACS), where the classifier can be trained accordingly.
This paper introduces a straightforward prototype for detection of drowsiness, where the Viola-Jones method is used for face recognition and cascade classifier is used for the detection of a contiguous sequence of eyes closed, which are considered as drowsiness.

***Index Terms*—drowsiness, cascade classifier, Viola-Jones method, FACS, AU**

***Resumen*—La detección y predicción de la somnolencia es clave para la implementación de vehículos inteligentes destinados a prevenir accidentes en carreteras. Existen varios enfoques para crear este tipo de vehículos. En este artículo se analiza el enfoque de visión por computador, donde dispositivos embebidos son usados conjuntamente con técnicas de inteligencia artificial y reconocimiento de patrones para implementar soluciones para la detección del nivel de fatiga de un conductor de un vehículo. La mayoría de investigaciones en este campo basados en visión por computador se enfocan en el análisis del parpadeo de los ojos del conductor, esta solución combinada con patrones adicionales como el reconocimiento del bostezo o el movimiento de la cabeza constituye ser una solución bastante eficiente. El primer paso en este enfoque es el reconocimiento del rostro, para lo cual el uso del algoritmo AdaBoost muestra resultados precisos en el proceso de extracción de características, mientras para la detección de somnolencia, el uso de clasificadores como el Support Vector Machine (SVM) muestra también resultados prometedores.**
Un componente básico en la tecnología de visión por computador es el uso de una base de datos de imágenes espontaneas acorde al Sistema Codificado de Acciones Faciales (SCAF), con la cual el clasificador puede ser entrenado. Este artículo presenta un prototipo sencillo para detección de somnolencia, en el cual el método de Viola-Jones es utilizado para el reconocimiento de rostros y un clasificador tipo cascada es usado para la detección de ojos cerrados en una secuencia continua de imágenes lo que constituye un indicador de somnolencia.

***Palabras clave*—somnolencia, clasificador tipo cascada, método Viola-Jones, SCAF**

D. Burbano is with the Swiss Joint Master of Science in Computer Science Program, University of Bern, University of Neuchâtel and University of Fribourg, Switzerland (e-mail: daniluvatar@gmail.com)

J. L. Carrera is with the Swiss Joint Master of Science in Computer Science Program, University of Bern, University of Neuchâtel and University of Fribourg, Switzerland (e-mail: jlcarvi@hotmail.com)

This paper was submitted to the LAJC on February 2015

I. Introduction

THERE are several human factors that causes highway crashes and vehicle collisions, one of them is driver drowsiness or fatigue. With the active intelligent vehicles research and last advances in embedded devices, the implementation of human activity recognition mechanisms aimed to prevent such accidents are becoming a subject of big interest, because it will transform positively the way drivers interact with their vehicles.

Automatic driver fatigue detection with embedded devices is one technology that can be used to achieve this, which is the main subject of this paper. Driver fatigue detection refers to the use of machine learning and pattern recognition techniques in order to determine the driver's fatigue level, whereas embedded devices refers to sensors used to gather and monitor the required data from the vehicle or the driver. By integrating these techniques, enough information for an accurate estimation of the driver's state is obtained.

A. Driver Fatigue Detection and Prediction

The mechanisms used to recognize driver fatigue, should focus not only on detecting but also on predicting driver drowsiness, so that they can really avoid potential vehicle collisions. There are some technologies identified as in [1], [2] which categorizes these solutions in four main groups described below:

1) Fitness for duty technologies: These technologies are intended to provide some behavioural or biological estimate of an operator's functional capability for work yet to be performed relative to a standard [2]. Thus most solutions are based on assessing the vigilance or alertness capacity of an operator before the work is performed. Performance of the subject at a chosen task is used as a measure to detect existing fatigue impairment. This approach involves eye hand coordination or driving simulator tasks methods, as well as sampling aspects of performance capability or physiological response.

Fitness for duty solutions are potentially good for measuring existing fatigue, but have rather dramatic learning curves,

additionally they are aptitude and language skill sensitive making their predictive validity still not well known.

2) Mathematical models of alertness dynamics: This approach uses mathematical models in order to predict the performance of an individual based on past sleep and workload factors. For instance an integration of the model is into a wrist-activity monitor and recorder which will store up records of the wearer's activity and sleep obtained over several days. This models show potential to easily predict fatigue in drivers but a large amount of validation and possible fine tuning of the models are needed before they can be fully accepted.

3) Vehicle-based performance technologies: These technologies are directed at measuring the behaviour of the transportation hardware systems under the control of the driver. Vehicle-based performance technologies places sensors on standard vehicle components, e.g., steering wheel, gas pedal, and analyses the signals sent by these sensors to detect drowsiness, under the assumption that it reflects identifiable alterations when the driver is fatigued. This approach is clearly a non-intrusive one, which is an advantage, but work in very limited situations and its implementation is complex.

4) In-vehicle, on-line, operator status monitoring technologies: These set of techniques are aimed to measure the behaviour of the driver. Several types of measurements are employed for acquiring the required data like video of the face (eyelid position, eye blinks, eye movements, pupil activity, facial tone, direction of gaze, head movements), eye trackers, wearable eyelid monitors, head movement detector, EEG[1] and ECG[2] devices. There are mainly two approaches used under this context:
i) Physiological Signals, ii) Computer Vision Systems. The former focus on the measurement of physiological signals such as heart rate, pulse rate and EEG. However this method has drawbacks in terms of practicality since it requires a person to wear an EEG cap while driving. The latter is a prominent technology in monitoring the operator status and a non-intrusive one, the remaining sections of this paper as well as our prototype will be based on this approach.

In this paper a description of the steps used for the implementation of a small prototype which detects fatigue and drowsiness using the computer vision approach will be made, where the Cascade classifier will be used for the face detection and drowsy driver detection as well.

II. Related Work

The studies done for detection of fatigue and drowsiness using the computer vision approach are focused mainly in the analysis of blinks, but there are also research works that incorporate additional facial expressions or human behaviour for drowsy driver detection.
Previous works as in [4] implemented a semi-automated eye tracking system that analyses measures associated with slow eye closure, based in video monitoring and applying a scientifically supported method known as PERCLOS (Percent Eyelid Closure). Some of these works used infrared cameras to estimate the PERCLOS measure. It is worth pointing out that infrared technology for PERCLOS measurement works fairly well in the darkness of night, but not very well in daylight, because ambient sun light reflections make it impractical to obtain retinal reflections of infrared [1]. Subsequent researches as in [5] used enhanced versions of the PERCLOS cameras, known as Copilot the second generator of PERCLOS monitor, which uses a structured illumination approach to identifying a driver's eyes.
Computer vision researches not only focus on blink rate for drowsiness, some studies as in [1] and [3] include eye closure, yawing and even head motion for the analysis. These works use machine learning techniques such as Adaboost [12] for the feature selection and SVM to detect the facial actions, achieving an accuracy of 96%. The studies shows significant associations between facial expression and fatigue beyond eye blinks, interesting findings like the fact that in 60 seconds before falling asleep, drivers yawn less, not more as is often thought, were done. This kind of studies are based upon important works as in [6] where detection of facial actions from spontaneous facial expressions were established through the implementation of FACS (Facial Action Coding System), this drove another studies for establishing suited datasets of FACS as [7] where a database of digitized images were constituted by performing multiple tokens of most primary FACS action units on several subjects of varying ethnicity. An interesting work for establishing a dataset of FACS is found in [11], where besides AU (Action Units), Lucey et al. define a set of Emotions in terms of these facial action units.
Considering a real time solution for drowsy driver detection requires in first place mechanisms for face detection of the driver, as in [8] which uses a probabilistic model learned using boosting methods, where the system is robust enough to detect differences in facial structure, including facial expressions and eyeglasses. Another interesting research as in [9] uses algorithms for face detection which are able to work in an unconstrained environment, where a modified Adaboost algorithm and Cascade classifier methods are used for a robust real time face detection.
The goal of this paper is to evaluate the cascade classifier used in Viola Jones algorithm to detect action units related to drowsiness in drivers. This is the first paper as far as we know, that evaluates the Viola Jones algorithm as a possible mechanism to detect drowsiness. The remaining sections of this paper are as follows: In section 3 a detailed overview of the Computer Vision Systems methods for drowsy driver detection along with FACS is presented, in section 4 the machine learning techniques used for the face and drowsiness detection are explained, finally in section 5 the implementation of the prototype is introduced along with its results and evaluation.

[1]EEG (Electroencephalography) is the recording of electrical activity along the scalp.

[2]ECG (Electrocardiography) is the recording of the electrical activity of the heart.

III. Computer Vision Systems

Computer vision is a prominent technology in monitoring the driver status. It is useful in detecting and recognizing the facial motion and appearance changes occurring during drowsiness. The advantage of computer vision techniques is that they are non-invasive, and thus are more amenable to use by the general public [1].

A. Face Detection

Face detection is a computer vision technology aimed to detect human faces features in an image ignoring anything else. There are several approaches to perform face detection, most of which deals with faces at arbitrary scales assuming upright faces [10]. One of the most used mechanisms for face detection is the Haar Feature-Based Cascade classifier [9], also known as Viola-Jones method. This approach combines four key concepts for detecting objects in images:

- Simple rectangular features, called Haar features.
- An Integral Image for rapid feature selection.
- The AdaBoost machine-learning method.
- A cascade classifier to combine many features efficiently.

Viola-Jones method is suitable for real-time recognition, because it is faster than any of their contemporaries [10].

B. Facial Action Coding System

Facial action coding system (FACS) are objective coding standards developed by behavioural scientists which captures the richness and complexity of facial expressions, which provides a general purpose representation that can be useful for many applications, thus it has been widely applied for computer vision systems aimed to recognize drowsiness or fatigue in drivers.

FACS provides an objective and comprehensive language for describing facial expressions, allowing the discovery of new patterns related to emotional or situational states. FACS has also been able to identify patterns of facial activity involved in alcohol intoxication that observers not trained in FACS failed to note [6].

FACS can be seen as a general database of coded human facial expressions composed of action units (AU) as shown in figure 1, where the application of computer vision mechanisms enabled the automatic coding of facial expressions providing data on the dynamics of facial behaviour at a resolution that was previously unavailable, making facial expressions measurable.

C. Workflow for FACS

Most of the implementations done for real time automated recognition of facial actions from the facial action coding systems as in [1] and [6], uses a process similar as the one illustrated in figure 2, where the first input is a video of the person, from which face and eyes detection algorithms are applied in real time, then the automatically detected faces are aligned based on the detected eye positions, cropped, scaled and then passed through a bank of Gabor filters using

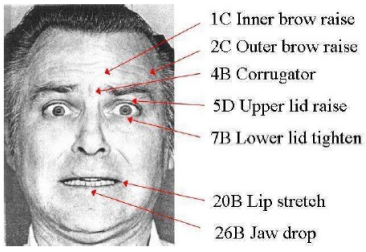


Fig. 1. Example facial action decomposition from the Facial Action Coding System with seven Action Units [6].

AdaBoost for the feature selection. Afterwards these outputs (i.e. filters selected by AdaBoost) are normalized and then passed to a standard classifier, like SVM which is trained for each of the AU (i.e. facial actions) which are considered as prediction for drowsiness, for instance the ones showed in table I.

The machine learning techniques used for automatic face and eyes detection, feature selection, and data-driven classifiers[3] could vary depending on the requirements. Barlett et al. [6] compared some machine learning algorithms using this workflow, and reported that the best results were obtained through the association of the techniques showed in figure 2. In this paper, the Cascade classifier will be used for the face detection and also used as the classifier trained for the AU considered as prediction for drowsiness.

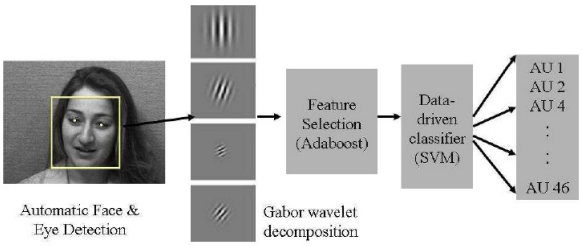


Fig. 2. Overview of fully automated facial action coding system [6].

IV. Machine Learning and Pattern Recognition Techniques

The techniques described in this paper are the ones used for the implementation of the prototype.

A. Viola-Jones Method

The basic principle of the Viola-Jones algorithm is to scan a sub-window capable of detecting faces across a given input image [9]. Viola and Jones use a classifier that, largely for the sake of computational efficiency, is based on an attentional cascade. The individual weak classifiers are based on a variant of the AdaBoost algorithm, which converts weak classifiers into a strong classifier via boosting. To be detected, an image must be detected by each level of a series of basic classifiers, each more discriminating than the last.

[3]data-driven classifiers describe the data to be matched

TABLE I
AN EXAMPLE OF A SET OF AU USED FOR PREDICTING DROWSINESS [1].

AU	Name
1	Inner Brow Raise
2	Outer Brow Raise
4	Brow Lowerer
5	Upper Lid Raise
6	Cheek Raise
7	Lids Tight
8	Lip Toward
9	Nose Wrinkle
10	Upper Lip Raiser
11	Nasolabial Furrow Deepener
12	Lip Corner Puller
13	Sharp Lip Puller
14	Dimpler
15	Lip Corner Depressor
16	Lower Lip Depress
17	Chin Raise
18	Lip Pucker
19	Tongue show
20	Lip Stretch
22	Lip Funneller
23	Lip Tightener
24	Lip Presser
25	Lips Part
26	Jaw Drop
27	Mouth Stretch
28	Lips Suck
30	Jaw Sideways
32	Bite
38	Nostril Dilate
39	Nostril Compress
45	Blink

The computational advantage is gained in the fact that the initial levels of the cascade can use very simple features for their classifiers, and therefore can reject the vast majority of locations in an image quickly [10]. There are three main steps to follow for applying this method, which are described below:

1) Integral Image: The first step of the Viola-Jones face detection algorithm is to turn the input image into an integral image. This is done by making each pixel equal to the entire sum of all pixels above and to the left of the concerned pixel [9], as is illustrated in figure 3, where positive and negative images are used to train the classifier.

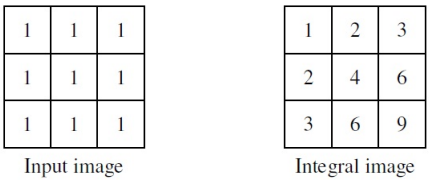


Fig. 3. The integral image [9].

2) Sum Calculation: The value of the sum of all pixels inside any rectangle can be done using only the values of the pixels that coincide with the corners in the input image, as is illustrated in figure 4.

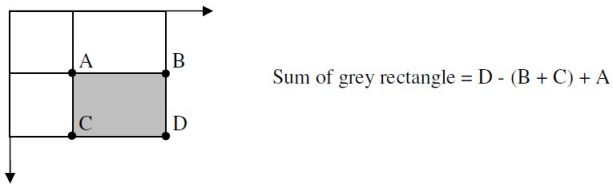


Fig. 4. Sum calculation into an integral image [9].

3) Feature Extraction: There are five types of features (reminiscent of Haar wavelets) defined, which consists of two or more rectangles. Each feature value is obtained by subtracting sum of pixels under the white rectangle from sum of pixels of black rectangle as illustrated in figure 5.
All possible sizes and locations are used to calculate features in the detector, this will result in over 160.000 features [9], where the best features are selected.

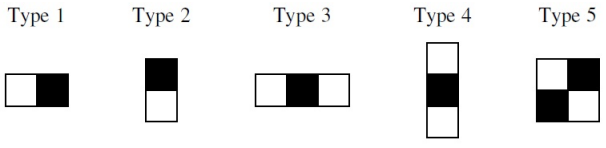


Fig. 5. Different types of features [9].

The tree steps stated above are all typically combined in a single schema so-called Scale Invariant Detector.

B. AdaBoost

AdaBoost is a machine learning boosting algorithm capable of constructing a strong classifier through a weighted combination of weak classifiers[4], where each feature is considered to be a potential weak classifier.

Applying to face detection, the goal of AdaBoost is to smartly construct a mesh of features capable of detecting faces. As stated above there can be calculated approximately 160.000 feature values within a detector at base resolution. Among all these features some few are expected to give almost consistently high values when on top of a face. In order to find these features Viola-Jones use a modified version of the AdaBoost algorithm. Since only a small amount of the possible 160.000 feature values are expected to be potential weak classifiers the AdaBoost algorithm is modified to select only the best features, where the best performing feature is chosen based on the weighted error it produces. This weighted error is a function of the weights belonging to the training examples.

As a result the weight of a correctly classified example is decreased and the weight of a misclassified example is kept constant, thus the second feature is forced to focus harder on the examples misclassified first [9].

C. Cascade Classifier

The cascaded classifier is composed of stages each containing a strong classifier. Applying to face detection, the job of each stage is to determine whether a given sub-window is definitely not a face or maybe a face. It is faster to discard a non-face than to find a face, therefore instead of finding faces, the algorithm should discard non-faces. With this in mind a detector consisting of only one strong classifier suddenly seems inefficient since the evaluation time is constant no matter the input. Hence the need for a cascaded classifier arises [9].

The approach is illustrated in figure 6, where features are grouped into different stages of classifiers and apply them one-by-one. Then if the detector fails in the first stage, this is discarded and the remaining features will be not considered.

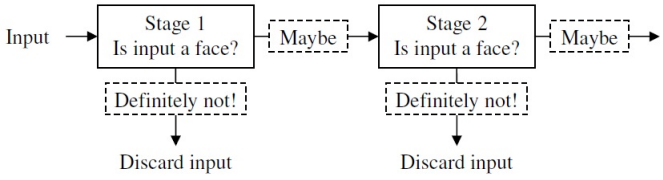


Fig. 6. The cascade classifier [9].

Regions of the image suspected to contain faces will have more attention.

The same principle is applied in the prototype for detection of drowsiness, where the AU “eyes closed” is discarded by each stage, as showed in figure 7.

[4]Weak classifiers classify correctly in only a little bit more than half of the cases.

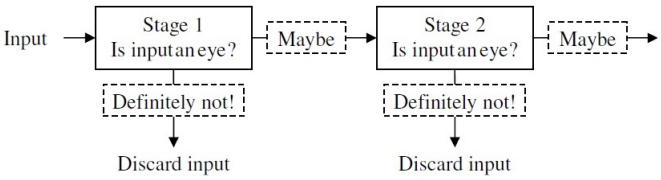


Fig. 7. The cascade classifier for the AU “eyes closed”.

V. PROTOTYPE

The main idea of the prototype is to simulate a real life scenario of a driver, thus the video camera of a computer is used to capture in real time the face of a person, and determine based on his eyes drowsiness. For achieving this, the prototype presents an automatic drowsy driver monitoring as showed in figure 8, where the system is based on tracking the changes in the eye blink duration by calculating eyes opening percentage (EOP) to launch an alarm when the system detects a drowsy driver.

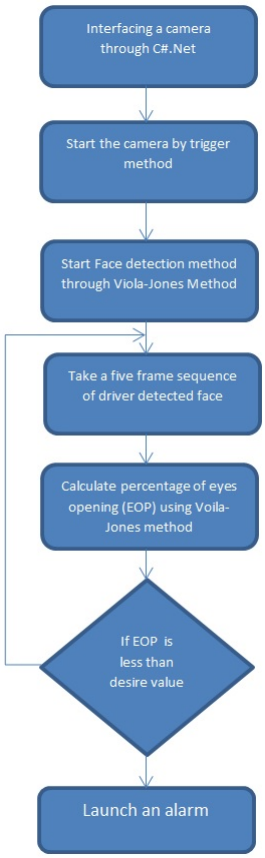


Fig. 8. Flow Chart of the Prototype

The prototype was done using the Cascade classifier for the face detection, based on XML descriptors which were built using AdaBoost frontal face detector, whereas for the detection of drowsiness the prototype uses a dataset of vector of features in XML format. Afterwards these vectors of features are also passed to a Cascade classifier, where one

TABLE II
ACTION UNITS USED FOR THE EVALUATION OF THE PROTOTYPE.

Eyes Opened	10639
Eyes Closed	69
Total	10708

TABLE III
RESULTS OF DETECTION OF EYES CLOSED USING CASCADE CLASSIFIER WITH 69 EXPRESSIONS WITH EYES ACTUALLY CLOSED.

Opened	10
Closed	59
Error rate	0.1449

TABLE IV
RESULTS OF DETECTION OF EYES OPENED USING CASCADE CLASSIFIER WITH 10639 EXPRESSIONS WITH EYES ACTUALLY OPENED.

Opened	8458
Closed	2181
Error rate	0.2050

straightforward AU "Eyes Closed" is used as a prediction for drowsiness.

A. Evaluation

An evaluation of the cascade classifier used in the prototype was done with a set of pictures, the algorithm was tested using the Extended Cohn-Kanade (CK+) database which consists of 593 expressions from 123 subjects with FACS [14], which include both posed an spontaneous expressions. For this prototype the AUs taken into account are shown in table II where it is possible to verify that a total of 10708 pictures where available to use for testing.

Table III shows the test results for the AU Eyes Closed, where the error rate is 14.49%, whereas the error rate for the AU Eyes Opened is 20.50% as shown in table IV. These result reflect that the algorithm has better accuracy for the AU eyes closed, nevertheless it would be advisable to test the results with more AU with eyes closed, to have a more realistic estimation, where according to this evaluation it would be expected to have an error rate of around 14%.

In order to assess the accuracy of the whole prototype, the precision was calculated using the formula 1, where the true positives and false positives are considered as measure, where the former is the number of times the algorithm recognised as opened the AU Eyes Opened or as closed the AU Eyes Closed, and the latter is the number of times the algorithm recognised as opened the AU Eyes Closed or closed the AU Eyes Opened. At the end the algorithm had an accuracy of 79.54% which is a reasonable precision for a small prototype, in addition this evaluation shows that the cascade classifier could be used for the recognition of AU such as eyes closed or eyes opened.

$$\frac{nbtruepositive}{nbtruepositive + nbtruenegative} \tag{1}$$

TABLE V
ACCURACY EVALUATION

Classifier	Recognition Rate
Bayes	75.57%
Neural Networks	76.89%
SVM	85.34%
Cascade Classifier	79.54%

Additionally the vectors of features generated from the feature extraction algorithm were used for training the most common classifiers.

$$x^{(AU)} = [Z_1, Z_2, Z_3, ..., Z_4] \tag{2}$$

By taking all the vectors as showed in 2, a dataset was built and applied directly to Weka for comparing the accuracy, as can be seen in table V.
Results show that Viola Jones method has more accuracy than Bayes and Neural Network machine learning algorithms, however the accuracy for SVM is better than Viola Jones but at the same time more complex to implement and it takes more computing resources to execute (i.e. computing expensive).

VI. CONCLUSION AND FUTURE WORK

Spontaneous expressions differ from posed expressions, thus a database of spontaneous facial expressions to train and evaluate systems for automatic recognition of facial expressions would be the best as it would give results closer to real life scenarios. The studies made for recognition of drowsiness based on facial expressions have more accurate results when are used along with another additional pattern like head motion. This fact shows that by including additional parameters like heart rate, head motion, voice tone, etc. for training the classifier, the results could be even better, thus as a future work, it would be very interesting to include such measures (using others embedded devices for gathering the required data) to assess the expectations of reduction in the error rate.
An algorithm for feature selection that is definitely suitable for evaluating and analysing human faces is AdaBoost, as showed in previous studies [6] this algorithm enhanced both speed and accuracy of SVM regarding the detection of a facial action (i.e. action unit), whereas for face recognition when AdaBoost is used in the Viola-Jones method the results are very efficient. The classifiers used for the task of facial action recognition can vary but the best results are with the data-driven ones.
Using emotions instead of just AU could be an interesting work to develop human activity recognition applications, where emotions such as angry, anxiety could trigger an alert for drivers, or recognise states like drunk, which could yield some instructions to the intelligent vehicle, so that it does not start working.

Based on the analysis and results of these studies, it would be possible to state that real applications of these technologies should have at least the characteristics below:

- A database fully FACS coded from spontaneous expressions.

- Using AdaBoost algorithm (or similar) for the feature extraction along with a data-driven classifier.
- Additional patterns to take into account like head motion, heart rate, etc.

Regarding the prototype, as a future work it would be also interesting to try to increase the accuracy not only by including additional parameters for training the classifier, but also to find out some variant of the cascade classifier algorithm and evaluate it with additional AU in an attempt to identify the ones with high accuracy for this classifier.

REFERENCES

[1] Esra Vural, Mujdat Cetin, Aytul Ercil, Gwen Littlewort, Marian Bartlett and Javier Movellan; *Automated Drowsiness Detection For Improved Driving Safety*

[2] David F. Dinges and Melissa M. Mallis; *Managing Fatigue by Drowsiness Detection: Can Technological Promises be Realized?*, 1998

[3] Esra Vural, Mujdat Cetin, Aytul Ercil, Gwen Littlewort, Marian Bartlett and Javier Movellan; *Machine Learning Systems for Detecting Driver Drowsiness*

[4] Grace, R., Byrne, V., Bierman, D., Legrand, J.M., Gricourt, D., Davis, B., Staszewski, J., Carnahan, B.; *A Drowsy Driver Detection System For Heavy Vehicles*

[5] Richard Grace, Sonya Steward; *Drowsy Driver Monitor and Warning System*

[6] Bartlett, M., Littlewort, G., Frank, M., Lainscsek, C., Fasel, I., Movellan, J.; *Automatic recognition of facial actions in spontaneous expressions. Journal of Multimedia.*, 2006

[7] Kanade, T., Cohn, J., Tian, Y.; *Comprehensive database for facial expression analysis.*

[8] Fasel I., Fortenberry B., M.J.; *A generative framework for real-time object detection and classification.*, 2005

[9] Ole Helvig Jensen; *Implementing the Viola-Jones Face Detection Algorithm.*, 2008

[10] Andrew King; *A Survey of Methods for Face Detection*, 2003

[11] Patrick Lucey, Jeffrey F. Cohn, Takeo Kanade, Jason Saragih, Zara Ambadar; *The Extended Cohn-Kanade Dataset (CK+): A complete dataset for action unit and emotion-specified expression*, 2010

[12] Yi-Qing Wang; *An Analysis of the Viola-Jones Face Detection Algorithm*, 2013

[13] Rohan Putta, Gayatri N Shinde, Punit Lohani; *Real Time Drowsiness Detection System using Viola Jones Algorithm*, 2014

[14] University of Pittsburgh, Cohn-Kanade database [Online]. Available: http://www.pitt.edu/~emotion/ck-spread.htm

Danilo Burbano Acuña received his B.S. degree in systems engineering from National Polytechnic School, Ecuador in 2006. From 2006 to 2013 he was a system analyst, consultant and software developer in three different enterprises. Starting in a banking software development company, then in a masive beverage production company and afterwards in a government company which controls and administrates the taxes around Ecuador. He is currently pursuing the M.S. degree in Computer Science with a specialization in Distributed Systems in a Swiss joint master program at the University of Bern, Neuchâtel and Fribourg. He has been coauthor of papers regarding the Internet of Things and is currently working in his master thesis about indoor localization and tracking. His research interest includes wireless sensor networks, localization of wireless devices and applications of artificial intelligence.

Jose Luis Carrera V was born in Quito, Ecuador. He received the B.S in systems engineering from National Polytechnic School of Ecuador in 2005 and his M.S. in Communication and Technologies of Information Management degree from National Polytechnic School in 2012. From 2005 to 2011, he was a geosystem analyst in National Geographic Institute of Ecuador where he worked in different projects with national scope. From 2011 to 2013, he was a Professor in the Computer Sciences Engineering Department in National Polytechnic School of Ecuador. Currently, he is pursuing another M.S. degree in Computer Science and Specialization in Distributed Systems in a joint master program at the University of Bern, Neuchâtel and Fribourg in Switzerland. His research interest areas include distributed systems with wireless sensor networks, mobile communications and pattern recognition and machine learning for human activity.

Modeling the Performance of MapReduce Applications for the Cloud

Iván Carrera and Cláudio Geyer

***Abstract*—In the last years, Cloud Computing has become a key technology that made possible to run applications without needing to deploy a physical infrastructure. The challenge with deploying distributed applications in Cloud Computing environments is that the virtual machine infrastructure should be planned in a time and cost-effective way.**

This work is a summary of a previous work presented by the authors as a Master's thesis, with the goal of showing that the execution time of a distributed MapReduce application, running in a Cloud computing environment, can be predicted using a mathematical model based on theoretical specifications. This prediction is made to help the users of the Cloud Computing environment to plan their deployments, i.e., quantify the number of virtual machines and its characteristics. After measuring the application execution time and varying parameters stated in the mathematical model, and after that, using a linear regression technique, the goal is achieved finding a model of the execution time which was then applied to predict the execution time of MapReduce applications. Experiments were conducted in several configurations and showed a clear relation with the theoretical model, revealing that the model is in fact able to predict the execution time of MapReduce applications. The developed model is generic, meaning that it uses theoretical abstractions for the computing capacity of the environment and the computing cost of the MapReduce application.

***Index Terms*—MapReduce, Cloud, Hadoop, FLOPs, MRBS, Performance**

***Resumen*—En los ltimos aos, Cloud Computing se ha convertido en una tecnologa clave que ha hecho posible ejecutar aplicaciones sin la necesidad de utilizar una infraestructura fsica. El desafo de implementar aplicaciones distribuidas en ambientes de Cloud Computing es que la infraestructura de mquinas virtuales debe considerar aspectos relacionados con el costo y el tiempo de utilizacin. Este trabajo es el resumen de uno anterior, presentado por los autores como tesis de maestra, con el objetivo de demostrar que el tiempo de ejecucin de una aplicacin distribuida MapReduce, ejecutndose en un ambiente de Cloud Computing, puede ser predicho utilizando un modelo matemtico basado en especificaciones tericas. Esta prediccin se realiza para ayudar a los usuarios de un ambiente de Cloud Computing a planificar sus implementaciones, es decir, cuantificar el nmero de mquinas virtuales y sus caractersticas. Despus de medir el tiempo de ejecucin de las aplicaciones y variando los parmetros establecidos por el modelo matemtico, y seguidamente usando una tcnica de regresin lineal, el objetivo se alcanza al encontrar un modelo del tiempo de ejecucin que fue posteriormente aplicado para aplicaciones MapReduce. Los experimentos fueron realizados en diferentes configuraciones y mostraron una clara relacin con el modelo terico, mostrando as que el modelo es capaz de predecir el tiempo de ejecucin de aplicaciones MapReduce. El modelo desarrollado es genrico, es decir que usa abstracciones tericas para la capacidad de cmputo del ambiente y el costo computacional de la aplicacin MapReduce.**

***Palabras clave*—MapReduce, cloud, hadoop, FLOPs, MRBS, rendimiento**

I. Introduction

Cloud Computing, as defined in [1] by the U.S. National Institute of Standards and Technology NIST, is a computing model that enables ubiquitous, on-demand network access to a shared pool of configurable computing resources. Likewise, the U.S. Department of Energy, DoE, says in [2] that a Cloud Computing model still lacks a good performance management, even though it can bring to the general user a big gain in terms of scalability, and usability of a computing infrastructure.

MapReduce is a programming model and an associated implementation for processing and generating large data-sets proposed by Google [3]. MapReduce performance depends, amongst other things, on the type of data that is going to be processed, so various types of workloads can have different characteristics.

Research presented in [4] establishes a number of parameters that have to be configured by the system administrators of a MapReduce application, in a way that is time-effective. As it is discussed in works as [5] and [6], an optimal cluster size should save money by optimizing infrastructure Cloud resources for the MapReduce user and the Cloud provider. This optimization can let users to take a better advantage of the resources they are using in the Cloud.

The research in this paper is the summary of a Master's thesis presented by the authors in [7], and motivated by the need of being able to predict the execution time of a distributed application running in a Cloud Computing environment, that would allow users to perform Capacity Planning, and so plan their virtual clusters in a way that is cost and time-effective. The main objective of this paper is to present a formula for predicting the execution time of a MapReduce application. This research's main approach is to develop a prediction model as was described in [8].

The prediction will be done with a formula that computes the execution time t of the application as a function of four variables:

1) W, the amount of workload;
2) p, the number of virtual machines the application will be run on top of, since it is a distributed application;
3) A, the type of the MapReduce application, since Map Reduce is a very versatile framework and many different

Iván Carrera. Department of Informatics and Computer Science National Polytechnic School (EPN). Isabel La Católica s/n, Quito, Ecuador. ivan.carrera@epn.edu.ec

Cláudio Geyer. Institute of Informatics (INF). Federal University of Rio Grande do Sul (UFRGS). Av. Bento Gonçalves, 9500, Porto Alegre, Brazil. geyer@inf.ufrgs.br

applications can be programmed with it. *A* will be expressed as a relation between the number of operations per GB the application has to perform in order to process the *W* amount of data in Map and Reduce phases; and,

4) *T*, the capacity of the *p* computers that compose the cluster on top of which the application will be run. *T* will be expressed in terms of the number of operations per second that every computer in the cluster is capable of perform.

Thus,

$$t = f(W, p, A, T) \quad (1)$$

This work is intended to help users to know *'a priori'* how much time it will take for the application to run, so they can plan their virtual machine clusters, and find a best-suited configuration for their MapReduce application.

The experimental results of this work are a valuable contribution, showing that it is possible to model the execution time of a MapReduce application, and how the performance of a MapReduce application in terms of execution time changes when running in different environment configurations.

The remainder of this paper is organized as follows: section II talks about the related research works about modeling the performance of MapReduce applications, section III addresses the theoretical mathematical model used for this work, section IV explains the performed experiments, how they were designed and executed, section V discusses the experimental results and how they prove our theoretical approach, and finally, section VI talks about the Conclusions of this work and suggests the future work for further research.

II. Related Work

Extensive and comprehensive models for each phase of MapReduce are presented in [9]. In said work, the MapReduce algorithm has two sets of tasks: Map, run first, and Reduce, run last. The Map task execution is divided into five phases: Read, Map, Collect, Spill and Merge; and in the same way, the Reduce task execution is divided into four phases: Shuffle, Merge, Reduce and Write. In [9], the execution time is calculated as a function of three parameters: *d*, data properties, *r*, cluster resource properties, and *c*, configuration parameter settings. It also shows a set of formulas that are useful to determine the execution time of a MapReduce job, introducing the concept of 'cost' for the stages, understood as what determines the amount of performed operations on the CPU or the hard drive to process data or write/read data from/to the hard drive. The Technical Report does not go further and does not assess the models with experimentation. The models are exclusively theoretical.

Another interesting work in the topic of MapReduce and Performance Evaluation is [10]. In said work, authors present three cost functions that show a relationship between some characteristics of the MapReduce application and the time that takes for the application to execute. These three cost functions for MapReduce differ from each other by taking into account more or less complexity of the MapReduce application; for example, the simplest approach assumes that the amount of input data is fixed, so it is not necessary to take into account in the cost function. Authors assess their performance model taking into account specific parameters of MapReduce applications like the number Map and Reduce slots, the number Map and Reduce rounds, the complexity of Map and Reduce phases and the cost of scheduling all said processes.

Other work showing performance models of MapReduce is [11], where MapReduce is modeled in three separated phases: Map, Shuffle and Reduce. Also, said work takes into account three parameters: *memory*, understood as the capacity of a cluster node to save information in its hard disk, *machine*, understood as the total number of nodes composing the cluster and *time*, as the running time available for the execution of the MapReduce application. Authors describe also the execution of a MapReduce job subject to a probability function of correctness, meaning that the job will be executed with no error in only some cases, defined by the probability function.

Authors in [12] determine 5 design factors of a cluster utilized to run MapReduce application that affect the performance, namely:

1) **I/O mode.** Which is the way the MapReduce application gets its input data: *direct* mode, when reading directly from the hard drive or *streaming* mode, when streaming by a communication scheme as TCP/IP or JDBC,
2) **Indexing.** Even if MapReduce is typically used for unsorted data, when using sorted files, or database indexed tables, performance seems to improve,
3) **Data parsing.** If there is any decoding procedures inside the Map phase that can be fixed or variable along the execution,
4) **Grouping schemes** of input data, and
5) **Block-level scheduling.** Showing that the scheduler performs faster when using larger blocks.

Authors also investigated alternative implementation strategies for each factor, and how they affect the general performance of MapReduce applications. They have evaluated the performance of MapReduce with representative combinations of these five factors using a benchmark consisting of seven tasks. Amongst their findings is that MapReduce achieves elastic scalability through block-level scheduling.

III. Proposed Model

Hadoop [13] is a Java implementation of MapReduce proposed by the Apache Foundation. It is the most used MapReduce implementation [14]. Hadoop's API (Application Programming Interface) allows the user to program the Map and the Reduce functions.

Hadoop was originally thought to be run as a distributed application on top of a cluster of homogeneous machines, sharing a distributed filesystem. Hadoop uses a distributed filesystem called the Hadoop Distributed FileSystem HDFS, which is similarly an open implementation of the Google FileSystem GFS described in [3]. HDFS performs similarly to Google File System [15]. HDFS is based on NFS, working as a shared partition, accessible from all machines, in every hard disk of the nodes composing the MapReduce cluster.

The Hadoop execution algorithm is based on the MapReduce proposed by Google. According to the Hadoop execution

algorithm [13], represented in figure 1 [14], it has four defined phases:

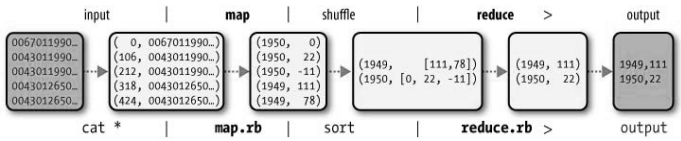


Fig. 1. MapReduce execution flow

1. Data distribution. The first process is to copy the data to the HDFS. This is done by the **master** node. The time length of this step depends on three factors: (1) the amount of data that has to be copied to the cluster nodes, (2) the number of nodes, and (3) the speed of the network that connects said nodes. Normally, all nodes would receive approximately the same amount of data workload, because the approach used is write-once read many.

2. Map phase. The second process is mapping the data, taking the input data and generating an output composed of *<key, value>* pairs according to the programming of the Map phase. Each node processes the data that it has locally stored. Since the Map phase is run by all the nodes in parallel, the time length of this phase would depend on the amount of workload a single node has to process, the speed of said node to process said individual workload, and the number of operations that involve the processing of the individual workload.

3. Shuffle. Once the Map phase has reached a 5% of its progress, map outputs are sorted to ease the processing in the next phase. Output data from the Map phase is transmitted over the network to the data nodes according to its content. The time length of this phase depends on the amount of data is transmitted and the speed of the network. This phase runs in parallel with the Map phase, and for matters of timing, we can only takes into account the shuffling of the last map outputs.

4. Reduce phase. Finally, the reduce phase takes the output of the shuffle phase and process it according to its programming. The time length of this phase depends on the amount of data that is going to be processed, the number of operations that would be required to process the Reduce inputs, again, the speed of the node.

Following, we propose propose an equation that can quantify the execution time of a Hadoop application, based on the aforementioned Hadoop algorithm.

As said in section I, there are 4 parameters for the evaluation of the MapReduce application, namely:

1) *W*, the amount of workload the application is intended to process, expressed in units of storage;
2) *p*, the number of computers the application will be run on top of, since it is a distributed application, it is supposed to be run on top of a cluster of computers, expressed as an integer;
3) *A*, the type of the MapReduce application, since MapReduce is a very versatile framework and many different applications can be programmed with it. *A* will be expressed as a the number of operations per unit of storage performed by the application in Map and Reduce phases; and,
4) *T*, the processing capacity of the p computers that compose the cluster on top of which the application will be run. *T* will be expressed in terms of the number of operations per second that every machine in the cluster is capable of perform.

In order to build a mathematical model of the MapReduce execution time, we have to make some considerations:

- The *Data Distribution* phase is not considered inside the execution time of the Hadoop program, so its duration can be disregarded;
- The *Shuffle* phase starts when the Map phase has reached a 5% of its completion, and runs simultaneously with the Map phase;
- For matters of simplifying the model, the *Shuffle* phase time will be considered only as the time when the *Map* phase has already finished;
- The *Reduce* phase starts only when the *Shuffle* phase has been fully completed;
- The *Map* and *Reduce* phases run in *slots*, that means that each node can run more than one Map or Reduce task in parallel; usually, the number of slots is related with the number of cores a node has.

Thus, the execution time of a Hadoop program can be modeled as the sum of the times of all phases:

$$t_{total} = t_{MAP} + t_{SHUFFLE} + t_{REDUCE} \tag{2}$$

The time of the Map phase t_{MAP} can be computed as the product of the time of a single Map task times the number of Map tasks per node:

$$t_{MAP} = t_{U_{MAP}} \cdot \frac{n_{MAP}}{p} \tag{3}$$

The time of a single Map task $t_{U_{MAP}}$ is computed as the product of the amount of workload for a single Map task W_{MAP} (known as a *chunk size*) times the cost of a single Map task $C_{U_{MAP}}$ (expressed as a relation of the number of floating point operations FLOPs required by the Map task and the chunk size), and divided by the capacity *T* of a single node (the number of floating point operations FLOPs per second):

$$t_{U_{MAP}} = \frac{W_{MAP} \cdot C_{U_{MAP}}}{T} \tag{4}$$

The number of Map tasks for a job is computed as the division between the total Workload and the *chunk size*:

$$n_{MAP} = \left\lceil \frac{W}{chunksize} \right\rceil \tag{5}$$

The time of the Reduce phase t_{REDUCE} can be computed as the product of the time of a single Reduce task times the number of Reduce tasks per node:

$$t_{REDUCE} = t_{U_{REDUCE}} \cdot \frac{n_{REDUCE}}{p} \tag{6}$$

The time of a single Reduce task $t_{U_{REDUCE}}$ is computed as the product of the amount of input workload for a single Reduce task $W_{U_{REDUCE}}$ times the cost of a single Reduce task $C_{U_{REDUCE}}$, and divided by the capacity *T* of a single node (the number of floating point operations FLOPs per second):

$$t_{U_{REDUCE}} = \frac{W^{IN}_{U_{REDUCE}} \cdot C_{U_{REDUCE}}}{T} \tag{7}$$

Replacing respectively equations 4 and 7 into equations 3 and 6 we have:

$$t_{MAP} = \frac{W_{MAP} \cdot C_{U_{MAP}}}{T} \cdot \frac{n_{MAP}}{p} \quad (8)$$

$$t_{REDUCE} = \frac{W^{IN}_{U_{REDUCE}} \cdot C_{U_{REDUCE}}}{T} \cdot \frac{n_{REDUCE}}{p} \quad (9)$$

Also, the time for the Shuffle phase $t_{SHUFFLE}$ can be computed as the product of the size of the output data of a single Map task times the number of remaining Shuffle tasks divided by the bandwith of the network:

$$t_{SHUFFLE} = \frac{W^{OUT}_{U_{MAP}} \cdot (n_{MAP} mod p)}{B} \quad (10)$$

The size of the output data of a single Map task $W^{OUT}_{U_{MAP}}$ can be computed as the division between the total amount of output data of the Map phase divided by the number of Map tasks.

$$W^{OUT}_{U_{MAP}} = \frac{W^{OUT}_{MAP}}{n_{MAP}} \quad (11)$$

So, replacing equations 8, 9 and 10 in equation 2, and simplifying W^{IN}_{REDUCE}, we have:

$$t_{total} = \frac{W_{MAP} \cdot C_{U_{MAP}}}{T} \cdot \frac{n_{MAP}}{p} + \frac{W^{OUT}_{MAP} \cdot (n_{MAP} mod p)}{n_{MAP} \cdot B} + \frac{W^{IN}_{U_{REDUCE}} \cdot C_{U_{REDUCE}}}{T \cdot p}$$

Equation III will serve as the performance model for MapReduce applications as a function of the parameters of the distributed system.

IV. Experiments

A. Scenarios

In order to verify the model of equation III, it is required that experiments are performed in several different environments, comprising private and public infrastructure, physical and virtual cloud infrastructure. The environments where the tests were run in, are described following:

1) **Cluster `gradep`**
   The `gradep` is a computer cluster on premises of Institute of Informatics *INF*, in the Federal University of Rio Grande do Sul *UFRGS*, composed of 18 nodes in total. Each node has an Intel Pentium 4 2.79 GHz CPU with 2 GB in RAM, and a Gigabit Ethernet connection. Each time the applications were run, the worker nodes were randomly chosen amongst these 18 machines to avoid errors produced by running all times in the same cluster nodes.
2) **Amazon EMR**
   Amazon Elastic MapReduce EMR [16] is a web service from Amazon Cloud services that uses Hadoop to process data across a cluster of Amazon EC2 instances [17].
   For the purposes of this research work, the `m1.small` instance type was used. EC2 instance types are specified in [17]. The processing power of Amazon EC2 instances is expressed in ECU, which is a unit with the equivalent CPU power of a 1.0-1.2 GHz 2007 Opteron or Xeon processor as specified to Amazon EC2 documentation. The `m1.small` instance type, for example, has the processing power of 1 ECU.
   According to [18], the theoretical peak performance can be computed for different instances from the ECU definition: a 1.1 GHz 2007 Opteron can perform 4 flops per cycle at full pipeline, which means at peak performance one ECU equals 4.4 gigaflops per second.
3) **Windows Azure HDInsight**
   Windows Azure HDInsight [19] is a Hadoop-based service from Microsoft Azure for running an Apache Hadoop solution in a Cloud Computing environment. HDInsight uses the General Purpose Instances from Microsoft Azure.
   In this work we used the Large (A3) compute instance type. According to the Windows Azure documentation, the A3 compute instance type has a 4 cores 1.6 GHz processor, and 7GB in RAM.

B. Software description

Varying the applications in the tests help the model to identify when it cannot be useful, or if it only serves for a certain type of applications. The applications used for the tests are described following:

- In the `gradep` cluster, two applications, named `sort` and `wordcount`, were used. These applications form part of the `text-processing` benchmark from the MapReduce Benchmark Suite *MRBS* described in [20].
  As it is explained in [21], the `sort` application takes an input of text registers, and sorts them depending on its contents. And the `wordcount` application, as explained in [3], takes a text input and counts the occurrences of each word, resulting in a sorted list of words indicating how many times they appeared in the input text.
  The `wordcount` application is *CPU bound*, meaning that the execution time is mainly limited by the processing power of the CPU, and the `sort` application is *CPU bound* and *IO Bound* were used, meaning that the execution time is mainly limited by the CPU and the IO system.
- And finally, in the Cloud environments of *Amazon* and *Azure* it was used a log processing application, which basically is a text processing application similar to the ones used in the environments explained above.
  In 2013, a private company located in Brazil, asked the GPPD/INF research group for help to develop and test a MapReduce application to process large amounts of logs and to be run in a Cloud environment. The project supported the research presented in this dissertation by providing a use case for what it is intended to do, to predict the execution time of a MapReduce application running in a Cloud environment. The log processing application takes the log records and rearranges the contents of each text line, leaving the output slightly smaller than the output. The logs contain the same information but in a different order. More information about the application and the project can be found in [22].

C. Specification of Experiments

1) Evaluation Technique: In the present work we used two evaluation techniques, one for develop the performance model in equation III, and another one to assess said performance model. *Analytical Modeling* was first used as an evaluation technique, where, based on theoretical models of MapReduce, a formula to model its performance was developed. The second evaluation technique used in this work is *Experimental Measurement*. In this technique, the analytical/theoretical hypothesis is tested with experimentation.

2) Performance Metrics: As it has been discussed earlier in this work, and modeled in equation III, the performance metric used in the experiments is the execution time of the applications, considered from the beginning of the first Map operation, until the end of the last Reduce operation.

3) Workload: As said in [23], when the workload used in the experiments is a typical application, and it is applied to test a set of environments, it is called a benchmark.

4) Parameter Values: The values for the 4 parameters described in section I depend on the scenarios described in section IV-A.

Table I shows the used values in the different scenarios.

TABLE I
PARAMETER VALUES

Scenario	`gradep` cluster	AWS	Azure
W	{1, 5, 10, 20, 25} [GB]	{1, 5, 10, 20, 25} [GB]	{1, 5, 10, 20, 25} [GB]
p	{4, 8, 12, 16} [nodes]	{4, 6, 8} [nodes]	{4, 6, 8} [nodes]
A	text processing	log processing	log processing
T	`gradep`	m1.small	A3

In table I, *W*, the amount of workload is expressed in GB; *p*, the number of nodes is an integer; *A*, the type of the Map Reduce application is expressed as an application type, while in the formula this categorical value has to be changed for a numeric one in $[flops/GB]$; and *T*, the node capacity is expressed as the name of the cluster the node belongs to, or the instance type if it belongs to a Cloud environment, while in the formula this categorical value has to be changed for a numeric one in $[flops]$.

V. RESULTS

A. Private Cluster `gradep`

As shown in table I, in the `gradep` cluster, two applications were run to obtain and test the performance model. These applications were taken from the text-processing benchmark of MRBS [20]. Values for *W* and *p* in table I were arbitrarily established in function of the available nodes in `gradep` cluster. Each combination was run 20 times, for a total of 800 executions.

After the executions for the `sort` application, we noticed that the Map phase output W_{MAP}^{OUT} is proportional to the Map phase input W_{MAP}. This makes sense because the amount of Map output data would depend on the amount of Map input data, and the following relation can be established:

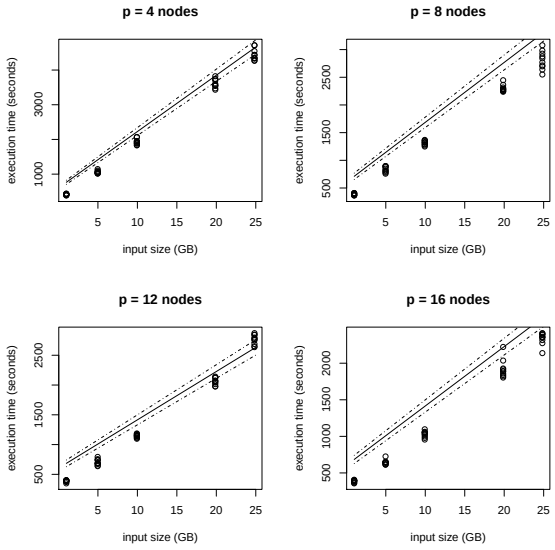


Fig. 2. Execution time vs. Workload for the `gradep` cluster

$$W_{MAP}^{OUT} \propto W_{MAP} \tag{12}$$

$$W_{MAP} = kW_{MAP}^{OUT} \tag{13}$$

And when replacing equation 13 into equation III, it simplifies to:

$$t_{total} = \frac{W_{MAP}}{pT} \cdot (C_{U_{MAP}} + k \cdot C_{U_{REDUCE}}) + \frac{W_{MAP}^{OUT} \cdot (n_{MAP} mod p)}{n_{MAP} \cdot B}$$

Without making these simplifications in equation III, we could not be able to calculate the values for *A*. Equation V-A shows the linear relation between the execution time t and the workload W_{MAP}.

Also, since it was required to quantify the computational power of the cluster nodes, that can be done using a benchmark. However, the tests were already performing a benchmark over the cluster, so, with the experimental data, it was possible compute the relation between the cost for Map and Reduce phases $C_{U_{MAP}} + k \cdot C_{U_{REDUCE}}$ and the computational power of the nodes T using the linear regression tool of the software *R* [24]. The value of the relation between the cost for Map and Reduce phases and the computational power was computed to:

$$\frac{C_{U_{MAP}} + k \cdot C_{U_{REDUCE}}}{T} = 6.05e^{-7} \left[\frac{s}{Byte}\right] \tag{14}$$

With an standard error of $SE = 1.15e^{-8}$ and a coefficient of determination of $R^2 = 0.88$. A value of $R^2 = 0.88$ gives us the confidence that the model is pretty close to an optimal fit, but considering this model being general and that it has to serve different values, it can be considered as a very good model.

In figure 2 it is shown the experimental data, grouped by the number of nodes used in each experiment and the corresponding line for the mathematical model with two auxiliary lines

TABLE II
RESULTS FOR MODEL ASSESSING ON THE GRADEP CLUSTER

Values	Predicted Value	Exp. Value	Error
W=1, p=6	401.6	430.7	7.24%
W=1, p=10	373.4	374.8	0.38%
W=1, p=14	339.7	387.3	14.0%
W=5, p=6	899.3	962.1	6.98%
W=5, p=10	752.9	734.7	2.42%
W=5, p=14	687.9	653.1	5.05%
W=10, p=6	1521.4	1442.0	5.22%
W=10, p=10	1227.3	1241.0	1.12%
W=10, p=14	1123.0	1250.3	11.33%
W=20, p=6	2765.6	2526.2	8.66%
W=20, p=10	2176.0	2454.4	12.8%
W=20, p=14	1993.3	2021.5	1.41%
W=25, p=6	3387.7	3392.4	0.14%
W=25, p=10	2560.3	2557.0	3.52%
W=25, p=14	2428.5	2296.9	5.42%

showing the 95% confidence interval of the model. Figure 2 shows that the experimental results follow a clear trend that is followed by the model.

After modeling the execution time, giving values to the formula of equation V-A, more experiments were run to assess the model, with the results detailed in table II.

Results in table II show that the model from equation V-A, using the values in equation 14, was actually able to predict the execution time of the MapReduce applications with a reasonable error. This error is small considering that when expressed in absolute value it reduces to a few minutes of execution. We have to keep in mind that these models are meant to be used in Cloud infrastructures, where a few minutes of difference do not mean excessive changes in the cost of using the platforms.

B. Amazon Elastic MapReduce

The experiments using the cloud infrastructure of Amazon Elastic MapReduce followed the values shown in table I. Each combination was run 10 times, for a total of 150 executions.

The main restrictions for only running 10 times was the budget and the fact that the account for running the tests in Amazon Web Services belonged to the enterprise that asked the GPPD/INF research group for assistance as it was explained in subsection IV-B and in [22].

The log processing application, as explained in section IV-B is an example of a sort application, meaning that the relation explained in equation 12 serves in this case as well. So, the model to follow with this environment is the one explained in equation V-A.

The value of the cost for Map and Reduce phases was computed using the software *R* [24] to:

$$C_{U_{MAP}} + k \cdot C_{U_{REDUCE}} = 2.16e^6 \left[\frac{flop}{Byte}\right] \qquad (15)$$

With an standard error of $SE = 6.46e^4$ and a coefficient of determination of $R^2 = 0.88$.

Following, in figure 3 it is shown the experimental data, grouped by the number of nodes used in each experiment and the corresponding line for the mathematical model with two auxiliary lines showing the 95% confidence interval of the model.

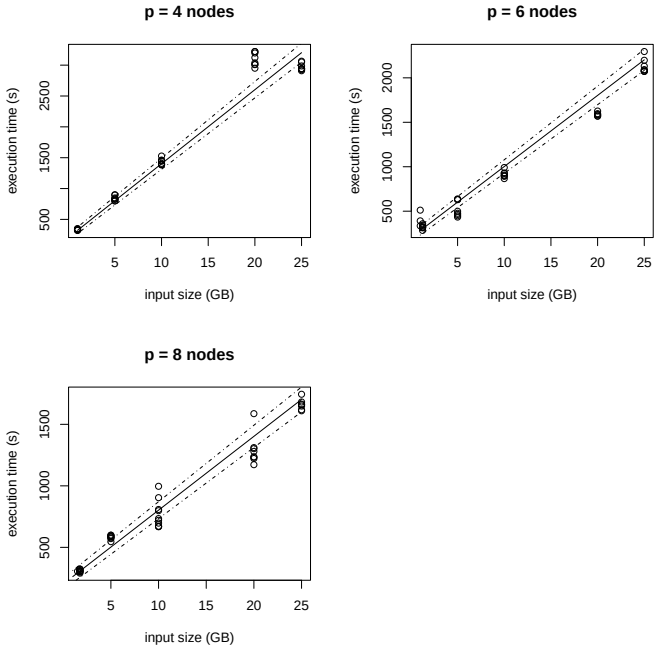


Fig. 3. Execution time vs. Workload for the Amazon EMR Environment

After modeling the execution time, giving values to the formula of equation V-A, more experiments were performed in order to assess the model, with the results described in table III:

TABLE III
RESULTS FOR MODEL ASSESSING OF THE AMAZON MEASURES

Values	Predicted Value	Exp. Value	Error
W=1, p=5	531.01	458.22	13.71%
W=1, p=7	488.23	361.53	25.95%
W=5, p=5	1061.4	870.24	18.01%
W=5, p=7	860.64	780.91	9.26%
W=10, p=5	1724.38	1556.93	9.71%
W=10, p=7	1326.15	1186.78	10.51%
W=20, p=5	3050.35	2826.88	7.33%
W=20, p=7	2257.17	2928.47	29.74%
W=25, p=5	3713.33	3556.38	4.23%
W=25, p=7	2722.68	2414.57	11.32%

Table III shows the percentage error from comparing the execution times obtained. Errors show a gap between experimental and theoretical values, however, considering that in a Cloud environment the execution times are charged by the hour or fraction, this values should be considered important only when reaching values close to one hour.

C. Windows Azure HDInsight

Experiments using the Windows Azure HDInsight cloud infrastructure used values from the table I. Each combination was run 10 times, for a total of 150 executions.

Similarly to the experiments in Amazon Elastic MapReduce, the main restrictions for only running 10 times each combination of parameter values were the budget and the fact that we were not the owners of the account for running the tests.

As in the previous case, the log processing application being an example of a sort application, follows the model explained in equation V-A. And, since there was no data to assign a value to T for the A3 Azure node type, the relation between the cost for Map and Reduce phases $C_{U_{MAP}} + k \cdot C_{U_{REDUCE}}$ and the computational power T was computed to:

$$\frac{C_{U_{MAP}} + k \cdot C_{U_{REDUCE}}}{T} = 2.16e^{6} \left[\frac{s}{Byte}\right] \qquad (16)$$

With an standard error of $SE = 1.06e^{-5}$ and a coefficient of determination of $R^2 = 0.86$.

In figure 4 it is shown the experimental data, grouped by the number of nodes used in each experiment and the corresponding line for the mathematical model with two auxiliary lines showing the 95% confidence interval of the model.

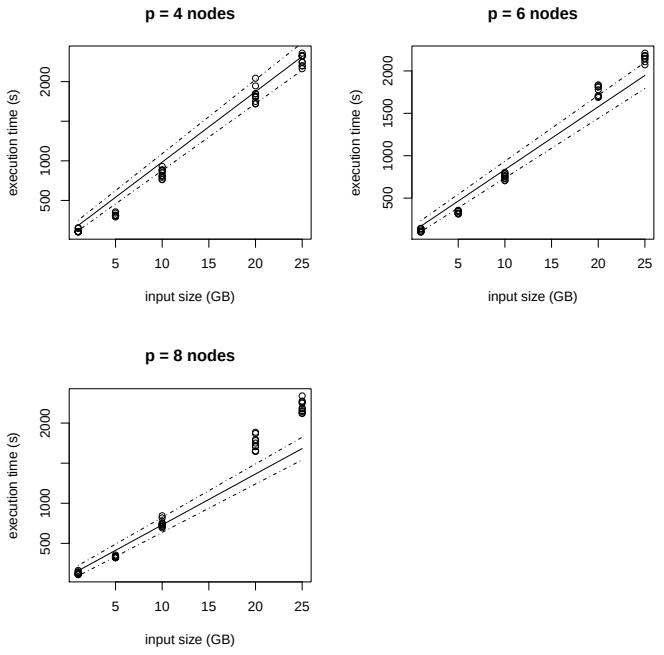


Fig. 4. Execution time vs. Workload for the Azure HDInsight Environment

After the execution time model was obtained, giving values to the formula of equation V-A, more experiments were run to assess the model and to verify that it was able to predict the performance of a MapReduce application running in the Azure HDInsight cloud environment. Each combination of W and p was run 10 times as well, in order to obtain the average and error values shown in table IV.

The values in table IV show the percentage error from comparing the execution times obtained. As in the previous cases, since experiments deal with commercial Cloud environments, where execution time is charged by the hour or fraction, this values should be considered big when reaching values close to one hour.

TABLE IV
RESULTS FOR MODEL ASSESSING THE MEASURES ON THE AZURE HDINSIGHT ENVIRONMENT

Values	Predicted Value	Exp. Value	Error
W=1, p=6	401.6	430.7	7.24%
W=1, p=10	373.4	374.8	0.38%
W=1, p=14	339.7	387.3	14.0%
W=5, p=6	899.3	962.1	6.98%
W=5, p=10	752.9	734.7	2.42%
W=5, p=14	687.9	653.1	5.05%
W=10, p=6	1521.4	1442.0	5.22%
W=10, p=10	1227.3	1241.0	1.12%
W=10, p=14	1123.0	1250.3	11.33%
W=20, p=6	2765.6	2526.2	8.66%
W=20, p=10	2176.0	2454.4	12.8%
W=20, p=14	1993.3	2021.5	1.41%
W=25, p=6	3387.7	3392.4	0.14%
W=25, p=10	2560.3	2557.0	3.52%
W=25, p=14	2428.5	2296.9	5.42%

As we can see from the results of tables III and IV, it is possible to predict the performance of a MapReduce application running in a Cloud Computing environment, thus achieving the main goal of this work.

VI. CONCLUSIONS AND FURTHER WORK

This research work was successful when predicting the execution time of MapReduce applications in the Cloud. A mathematical model with the form: $t = f(W, p, A, T)$ was developed and specified in some cases.

The mathematical model is based on the Hadoop Algorithm model described in section III. The formulae expressed in the same section was able to compute the execution time of MapReduce applications, for the cases described in section IV.

Results of section V show that there is an error rate when assessing the predictions made by the formula. It means that some further work, focusing in tuning the formula, should be done. Also, the shown simplifications of the formula allow us to see more easily the factors that have a direct impact on the execution time of the MapReduce applications.

The formula expressed in equation III is a suitable model for the execution time of MapReduce applications. Amongst its strengths we can say that it is a general model, since it does not depend on a type of application or hardware, it models the execution time with general parameters like the computing power of nodes, expressed as a number of operations per second performed by their processors, and the computing cost of Map and Reduce phases, expressed as the number of operations that the nodes have to process per unit of storage. This model should be able to compute the execution time of a large set of MapReduce applications.

In this work we have proven the idea explained in [8] where we said that following a performance evaluation methodology, we should be able to predict the execution time of a distributed application in a Cloud environment and, conversely, given a time constraint we could be able to know what virtual infrastructure can be enough to execute the distributed application.

Being able to predict the execution time of distributed applications in the Cloud can be useful when preparing a

budget. A MapReduce user can calculate the time and know how much time it will take to run the application.

As it was stated in the motivation of this work, further work is encouraged to test the approach described in this work with different distributed applications, and different scenarios. Also, further research can be done in order to improve the services of Cloud providers, namely:

From the point of view of a Cloud provider, being able to tell the time that a distributed application will take to run *'a-priori'* is useful in business models like the Spot Instances of Amazon [17]. In said business model, providers offer virtual machines with lower prices, based on the fact that they were reserved for other users for a longer time than the one they were actually busy. So, the provider can re-lease the virtual machines in lower prices. If a provider is able to know when a virtual machine will be freed, it can predict when it can be used as a Spot Instance.

With the models exposed in section V, a Cloud provider can use the approach explained in this work to program a feature of its Cloud platform where a Cloud user can know how much time a given MapReduce application will take to run, letting him program his budget.

Some other improvements in Cloud services can be performed, and hopefully this work can serve as a basis of a subject that could represent an important advance in Cloud services.

ACKNOWLEDGMENTS

This work was made with the support of the *Programa Estudantes-Convênio de Pós-Graduação (PEC-PG)*, of CAPES/CNPq - Brazil.

REFERENCES

[1] P. Mell and T. Grance, "The nist definition of cloud computing (draft)," *NIST special publication*, vol. 800, p. 145, 2011.

[2] K. Yelick, S. Coghlan, B. Draney, R. S. Canon *et al.*, "The magellan report on cloud computing for science," *US Department of Energy Office of Science, Office of Advanced Scientific Computing Research (ASCR) December*, 2011.

[3] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[4] S. Babu, "Towards automatic optimization of mapreduce programs," in *Proceedings of the 1st ACM symposium on Cloud computing*. ACM, 2010, pp. 137–142.

[5] H. Herodotou, F. Dong, and S. Babu, "No one (cluster) size fits all: automatic cluster sizing for data-intensive analytics," in *Proceedings of the 2nd ACM Symposium on Cloud Computing*. ACM, 2011, p. 18.

[6] R. Boutaba, L. Cheng, and Q. Zhang, "On cloud computational models and the heterogeneity challenge," *Journal of Internet Services and Applications*, vol. 3, no. 1, pp. 77–86, 2012.

[7] I. Carrera Izurieta and C. Geyer, "Performance modeling of mapreduce applications for the cloud," Master's thesis, Universidade Federal do Rio Grande do Sul, 2014. [Online]. Available: "http://hdl.handle.net/10183/99055"

[8] I. Carrera and C. Geyer, "Impressionism in cloud computing. a position paper on capacity planning in cloud computing environments," in *Proceedings of the 15th International Conference on Enterprise Information Systems (ICEIS)*. INSTICC, 2013, pp. 333–338.

[9] H. Herodotou, "Hadoop performance models. technical report cs-2011-05," *Duke Computer Science*, 2011. [Online]. Available: "http://www.cs.duke.edu/starfish/files/hadoop-models.pdf"

[10] F. Tian and K. Chen, "Towards optimal resource provisioning for running mapreduce programs in public clouds," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*. IEEE, 2011, pp. 155–162.

[11] H. Karloff, S. Suri, and S. Vassilvitskii, "A model of computation for mapreduce," in *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 2010, pp. 938–948.

[12] D. Jiang, B. C. Ooi, L. Shi, and S. Wu, "The performance of mapreduce: An in-depth study," *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 472–483, 2010.

[13] Hadoop, 2013, apache Hadoop https://www.grid5000.fr/ accessed on 12/28/2013.

[14] T. White, *Hadoop: the definitive guide*. O'Reilly, 2012.

[15] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*. IEEE, 2010, pp. 1–10.

[16] EMR, 2013, amazon Web Services - EMR Elastic MapReduce http://aws.amazon.com/elasticmapreduce accessed on 07/23/2013.

[17] EC2, 2013, amazon Web Services - EC2 Elastic Compute Cloud http://aws.amazon.com/ec2 accessed on 07/23/2013.

[18] A. Iosup, S. Ostermann, M. N. Yigitbasi, R. Prodan, T. Fahringer, and D. H. Epema, "Performance analysis of cloud computing services for many-tasks scientific computing," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 22, no. 6, pp. 931–945, 2011.

[19] HDInsight, 2013, windows Azure HDInsight http://azure.microsoft.com/en-us/documentation/services/hdinsight/ accessed on 12/02/2014.

[20] A. Sangroya, D. Serrano, and S. Bouchenak, "Benchmarking dependability of mapreduce systems," in *Reliable Distributed Systems (SRDS), 2012 IEEE 31st Symposium on*. IEEE, 2012, pp. 21–30.

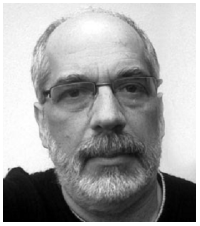
[21] O. OMalley, "Terabyte sort on apache hadoop," *Yahoo, available online at: http://sortbenchmark. org/Yahoo-Hadoop. pdf*, pp. 1–3, 2008.

[22] I. Carrera, F. Scariot, P. Turin, and C. Geyer, "An example for performance prediction for map reduce applications in cloud environments," in *Escola Regional de Redes de Computadores ERRC - RS Rio Grande do Sul*, 2013.

[23] R. Jain, *The art of computer systems performance analysis*. John Wiley & Sons Chichester, 1991, vol. 182.

[24] R, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2011, ISBN 3-900051-07-0 http://www.R-project.org/.

Iván Marcelo Carrera Izurieta is an assistant professor at the Department of Computer Science and Informatics of the National Polytechnic School. His research interest include parallel and distributed computing, performance evaluation and innovation. He received his MSc in Computer Science from the Informatics Institute of the Federal University of Rio Grande do Sul. He's a member of the Brazilian Computing Society. Contact him at Escuela Politécnica Nacional, Quito, Ecuador; ivan.carrera@epn.edu.ec.



Cláudio Fernando Resin Geyer is an associate professor at the Informatics Institute of the Federal University of Rio Grande do Sul. His research interests include ubiquitous computing, parallel and distributed computing, grid computing, and distributed objects. He received his PhD in informatics from the Joseph Fourier University. Hes a member of the ACM and the Brazilian Computer Society. Contact him at Universidade Federal do Rio Grande do Sul, Porto Alegre, Brazil; geyer@inf.ufrgs.br.

3D Sound applied to the design of Assisted Navigation Devices for the Visually Impaired

José Francisco Lucio Naranjo, Roberto Tanenbaum, Henry Patricio Paz Arias, Luis Alberto Morales Escobar and Carlos Efraín Iñiguez Jarrín

***Abstract*— This work presents an approach to generate 3D sound by using a set of artificial neural networks (ANNs). The proposed method is capable to reconstruct the Head Related Impulse Responses (HRIRs) by means of spatial interpolation. In order to cover the whole reception auditory space, without increasing the network complexity, a structure of multiple networks (set), each one modeling a specific area was adopted. The three main factors that influence the model accuracy --- the network's architecture, the reception area's aperture angles and the HRIR's time shifts --- are investigated and an optimal setup is presented. The computational effort to process the ANN is shown to be slightly smaller than traditional interpolation methods and all error calculation reached very low levels, validating the method to be used in the design of a 3D sound emitter capable of provide navigation aid for the visually impaired. Two approaches are presented in order to detect obstacles, one which makes use of computational vision techniques and other with laser proximity sensors.**

***Index Terms*—Acoustical Virtual Reality, Auralization, Artificial Neural Networks, HRIR Interpolation, ETA devices.**

***Resumen*— Este trabajo presenta un abordaje para generar sonido 3D utilizando un conjunto de redes neuronales artificiales (RNAs). El método propuesto es capaz de reconstruir la Respuestas Impulsivas Asociadas a Cabeza Humana (HRIRs) mediante interpolación espacial. Con el fin de cubrir todo el espacio de recepción auditivo, sin aumentar la complejidad de la red, fue adoptada una estructura de múltiples redes (conjunto), cada una modelando un área específica. Los tres factores principales que influyen en la exactitud del modelo --- la arquitectura de la red, ángulos de apertura de la zona de recepción y los cambios de tiempo del HRIR --- son investigados y es presentada una configuración óptima. El esfuerzo computacional necesario para procesar la RNA muestra ser menor que métodos tradicionales de interpolación y todos los cálculos de error alcanzan niveles muy bajos, validando el método para ser utilizado en el diseño de un emisor de sonido 3D capaz de proporcionar asistencia en la navegación de discapacitados visuales. Dos enfoques se presentan con el fin de detectar obstáculos, uno que hace uso de técnicas de visión computacional y otro con sensores de proximidad de láser.**

***Palabras clave*— Realidad Virtual Acústica, Aurilización, Redes Neuronales Artificiales, Interpolación de HRIR, Dispositivos ETA.**

I. INTRODUCTION

It is well known that navigation could represent a dangerous activity for the visually impaired. Nonetheless, the recent development of ETA devices (Electronic Travel Aid) provides means to detect obstacles (position, distance or even size). In such a case, the human hearing attribute can be one of the best ways to communicate a save path to walk avoiding obstacles. In this sense, virtual acoustic reality is the most powerful tool to consider for generating a 3D sound emitter.

Nowadays, acoustical simulation encompasses not only the assessment of acoustical parameters, such as levels and reverberation times, but accounts with a powerful tool: The auralization. It consists in generating the sound heard by a subject when immersed into a simulated environment, which can be a simple room, a somewhat complex theater, an industrial plant or even an urban space. This sound must be reproduced to be heard by a human being, in an environment free from sound reflections, i.e, in an anechoic chamber with a cross-talk cancellation system --- which is unavailable for most users --- or through an equalized headphone set.

The sound that arrives at the ears entrance is altered due to diffraction and absorption, among other phenomena [1], which is highly dependent on the wavefront incidence direction. The human head and torso constitute natural acoustic filters from every source position to the ears entrance. Such filters can be modeled as Finite Impulse Response (FIR) systems, known as Head-Related Impulse Responses (HRIRs). They are responsible to confer the 3D sound sensation in virtual

José F. Lucio Naranjo, Departamento de Informática y Ciencias de la Computación, Facultad de Ingeniería de Sistemas, Escuela Politécnica Nacional, Quito, Ecuador; (email: jose.lucio@epn.edu.ec)

Roberto A. Tenenbaum, Laboratório de Instrumentación en Dinâmica, Acústica y Vibraciones – LIDAV, Departamento de Modelagem Computacional, Instituto Politécnico, Universidade del Estado del Rio de Janeiro, Nova Friburgo, Brasil; (email: ratenenbaum@gmail.com)

Luis Morales is currently acting as titular professor at the Escuela Politécnica Nacional of Ecuador, and he is manager of Electronic Instrumentation Laboratory of the Faculty of Electrical and Electronic Engineering from the EPN. (email: luis.moralesec@epn.edu.ec)

Ing. Paz is currently acting as computer science titular professor at the National Polytechnic School (EPN) of Ecuador. (email: henry.paz@epn.edu.ec)

MSc. Carlos Iñiguez is acting as titular professor at Escuela Politécnica Nacional of Ecuador (EPN), and he is currently leading the software development for the research project ¨Simulation of Acoustic Wave Propagation in closed areas¨, at EPN. (email: carlos.iniguez@epn.edu.ec)

environments and to provide directional cues for human ability to distinguish the sound source locations [2].

Considering that auralization is a virtual reality process and that HRIR modeling is an important part of such systems, computational load is always a subject of concern. The most common solution consists in reducing the number of mathematical operations and/or simplifying the models involved in the auralization process. Nevertheless, such approach may cause a significant reduction of the 3D sound sensation and therefore compromise the receiver ability to recognize the sound source's direction.

Although, on real or simulated environments, the sound wavefront may come from anywhere and, even in the most complete HRIR databases, these functions are measured for discrete spatial locations. Such spatial discreteness leads to undesired audible effects, mainly when fast sound source movements are reproduced, producing 'clicks' or subtle changes [3]. In this case, two solutions are available: To round off the sound direction to the closest measured one; or to interpolate the closest functions to achieve a better estimative of such direction characteristics. Several techniques have been presented for HRIR interpolation [4, 5, 6, 7] while other were developed for continuous HRIR [8, 9, 10].

In a previous work, a method based on Artificial Neural Networks (ANNs) was introduced for interpolating HRIRs [11]. In that model an ANN committee is responsible for several reception areas around the listener, where each network models the functions (HRIRs) inside those areas.

The ANN main task is to synthesize an interpolated HRIR, receiving as input a vector with the direction of the desired sound source. This is accomplished thanks to a training procedure, in which several target functions are generated by using the bilinear interpolation technique (as shown in Fig. 1) and repeatedly presented to the network for learning purposes. In this approach, the interpolated (target) function is given by weighted sum of four known HRIR (measured ones):

$$\widehat{\mathrm{HRIR}}(n) = \sum_{i=1}^{4} \alpha_i \mathrm{HRIR}_i(n), \qquad (1)$$

where α_i are the weights for the measured $\mathrm{HRIR}_i(n)$ [12].

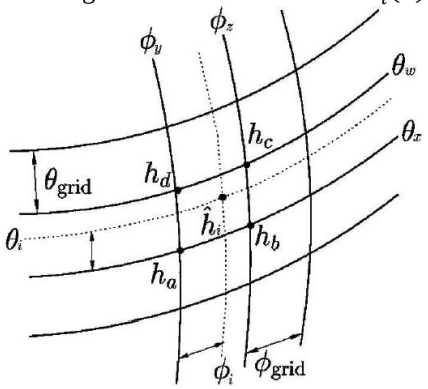


Figure 1: Bilinear interpolation scheme for a given direction coordinate *i*.

It is worth mention that the auralization procedure also involves a convolution product between an interpolated HRIR (corresponding to the sound source position) and an arbitrary anechoic signal. Nevertheless, this interpolation task constitutes an optimization that will facilitate the implementation of a 3D sound emitter, a constitutive part of the electronic travel aid (ETA) device for the visually impaired that is currently being developed.

In this work, the procedure to determine the optimal architecture and the reception area size, is presented. First, the model is presented, then the different steps to setup and to train the ANN to achieve the HRIRs interpolation capability are described. Then, the model performance is evaluated according to different error criteria. Finally, two approaches are presented in order to provide means to identify obstacles in which virtual sound sources will be simulated.

II. MODEL DESCRIPTION

An ANN is consisted by a set of simple processing elements, called artificial neurons, which have mutual influence behavior via a network of excitatory or inhibitory weights [13]. In order to define the optimal values of these synaptical weights (randomly initiated at first), a common supervised learning procedure is used. In such training procedure, which is called *backpropagation* [14], the ANN's weights are adjusted proportionally to their contribution to the observed error between the network output and the desired outputs (targets).

In this case, a *feed-forward multi-layer* ANN [15] was chosen to implement the HRIR interpolator system, due to the fact that the multi-layer feature provides means to optimize the ANN response by adjusting the network architecture (number of layers and number of neurons in each layer).

The network architecture, their number of inputs, outputs and internal layers depend on the data nature of excitation and output. Working as an interpolator, the input is the sound source direction, represented in a spherical coordinate system by two angular variables: The azimuth (ϕ) and the elevation (θ) angles. The desired outputs are the HRIR coefficients (time samples), whose targets are obtained from bilinear interpolation technique, as shown in Fig. 1.

The number of samples originally defined by the measurements conducted by Gardner and Martin [16] was 512 at a sampling rate of 44.1 kHz. However, most part of these samples tends to zero, due to the natural decay of the HRIR functions. Therefore, the most significant information is contained, approximately, in the first 100 samples [17]. Therefore, the number of samples chosen for the output functions was 128, since it facilitates any post signal processing with "power of two"' Fast Fourier Transform (FFT) algorithm.

The error used for synaptical weight optimization during the ANN's training is the mean squared error (MSE) between the target functions and the network output. The network architecture has an input vector with 2 elements (azimuth and elevation), one hidden layer with *L* neurons and the output layer with 128 neurons, as shown in Fig. 2.

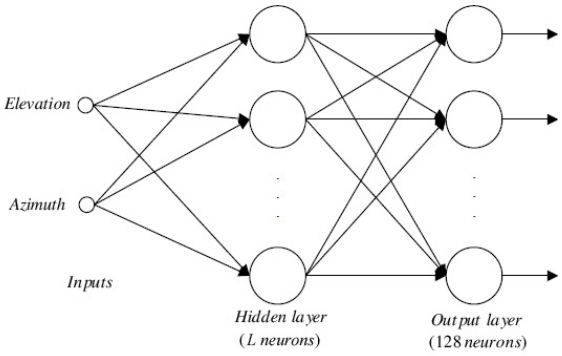


Figure 2: Network architecture for HRIR interpolation.

In order to improve the system performance, the listener surrounding space is split in several areas, as shown in Fig. 3. The subdivision area criteria is based on the tradeoff between network complexity and modeling error. Each region encapsulates several HRIRs, whose similarity depends on the aperture angle, i.e., wider regions present less correlated functions, especially for incidence direction where the sound suffers higher diffraction effects.

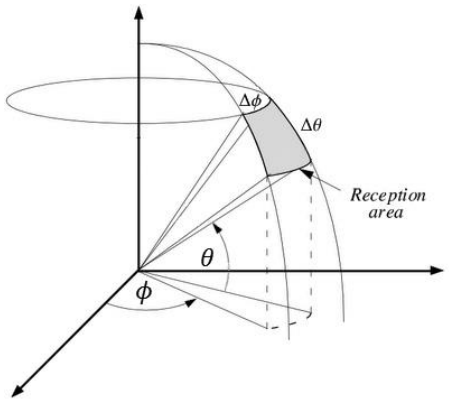


Figure 3: Network operation area limited in azimuth by ϕ and $\phi+\Delta\phi$ and in elevation by θ and $\theta+\Delta\theta$.

A similar research [18] uses just one network to cover a wide reception area. Although a complex architecture was used, the errors still were not small enough for an auralization processing. Therefore, it was assumed that smaller areas will facilitate the training procedure (producing smaller errors) and guaranteeing simple network architectures (single hidden layer with a few neurons for fast processing speed) due to the similarity of the functions involved. The numerical results will show if this assumption is correct.

Since each target function comes from an interpolation procedure whose input can be any arbitrary direction, it is possible to generate as many input/target pairs as needed. In order to cover the entire reception area, a grid distribution was applied to establish the position of the training parameters. The positions of the validation parameters were randomly chosen. All this taking care not to produce a pair where exist a measured HRIR, as shown in Fig. 4. The measured functions will be used later in order to test the accuracy of the RNA model.

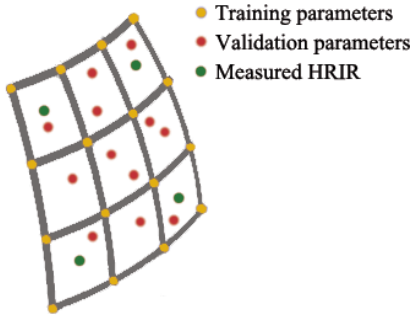


Figure 4: Input/target pairs distribution inside the reception area.

III. CRITERIA FOR PERFORMANCE EVALUATION

The model performance, including training error and target-output (mis)matching, is governed mainly by the following factors:

a. **Number of neurons in the hidden layer**: It defines the specificity in which a network will respond to training. An over-dimensioned number of neurons, besides demanding more computational effort to be processed, may lead the network to learn unimportant details of the training parameters, losing its generalization capacity for new parameters (overfitting). On the other hand, too few neurons in the hidden layer may decrease the network capacity to learn and, therefore, lose the output resolution.

b. **Reception area aperture**: When working with areas, the similarity of the HRIR inside such areas may contribute or degradate the network performance, according to the spectral characteristics of HRTFs, i.e., magnitude and phase, since the training is done in the time domain. Inside small areas there are only a few measured functions, which cause high similarity in the target functions, since they are derived from interpolation of very close locations. Besides, closer functions (HRIRs) present very small delays and highly correlated spectrum. On the other hand, wider areas, such as 90°, 180° or even 360° of azimuth, will include targets with very low similarity, both in spectrum and delays. Such differences require higher network complexity to deal with such variety of data. Therefore, the investigation of the aperture areas is a key element for the success of the proposed model. However, such similarity is not uniform around the listener space. It depends on the sound source location and may lead to a non-uniform space discretization.

c. **Time shifts**: The HRIRs present initial delays that

depend on the position of sound source relative to the head (ears). These time-shifts are caused by the distance between the sound source and the ears entrance. The presence of such delays may increase the network performance if the functions were highly correlated in terms of spectral characteristics (magnitude and/or phase). This occurs generally for smaller areas. However, for wider areas, these delays may decrease the network ability to learn patterns, since there are high spectral fluctuation between input samples and the time-shift are also larger due to the possibility of sources located far from each other. Therefore, for such situation, by removing these initial time delay, a more homogeneous set of data may aid the training process.

Therefore, it was observed that the influence of combined effects of the above parameters requires a more detailed analysis. As stated before, the MSE (training error) is not suitable for correctly evaluating the network performance under the human hearing point of view. In this section, one error criteria is presented in order to evaluate the accuracy of the proposed interpolation system.

A. Mean Magnitude Absolute Error (MAE)

The Mean Magnitude Absolute Error is a difference measure between the magnitudes from target and output functions. It is a scalar computed for a given direction. Equation (2) presents the average of the absolute differences between magnitudes.

$$\mathrm{MAE}(\theta,\phi) = 10\log\left(\frac{1}{\Omega}\sum_{\omega=1}^{\Omega}\left|\mathrm{M}_{\theta,\phi}(\omega) - \widehat{\mathrm{M}}_{\theta,\phi}(\omega)\right|\right), \quad (2)$$

where $\mathrm{M}_{\theta,\phi}(\omega)$ is the magnitude of the frequency response $\mathrm{H}_{\theta,\phi}(\omega)$ of the target function, given by

$$\mathrm{M}_{\theta,\phi}(\omega) = \left|\mathrm{H}_{\theta,\phi}(\omega)\right|, \quad (3)$$

while $\widehat{\mathrm{M}}_{\theta,\phi}(\omega)$ is the same for the output function for a given direction (θ,ϕ) and Ω is the number of discrete frequency bins used in the FFT.

IV. MODEL PERFORMANCE ANALYSIS

In order to determine the system behavior as a function of the mentioned parameters, several networks were trained for different configurations. Networks were trained from aperture angles varying from $5° \times 5°$ ($\Delta\phi \times \Delta\theta$, azimuth and elevation) to $40° \times 40°$, covering the whole auditory listener space. For each reception area's size (a network's operation area), the train parameters distribution density was kept constant.

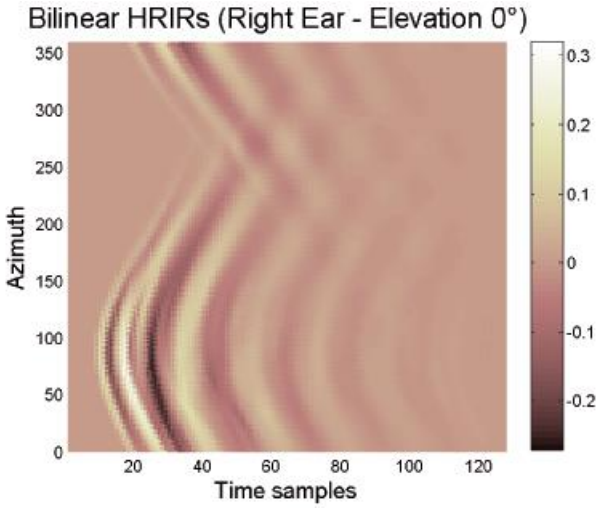
Also, for every trained area, two situations were considered: HRIR functions with time-shifts (original measured delays) and functions whose delays were removed. In such case, the delay were removed according to the assumption of a spherical shape head, where the distance $\mathrm{D}_{\theta,\phi}$ traveled by the sound wave were calculated considering the duplex theory [19], stated below.

$$\mathrm{D}_{\theta,\phi} = 2r\cos^{-1}(\cos(2\pi\theta)\cos(2\pi|\phi - a|)), \quad (4)$$

where θ and ϕ are the elevation and azimuth of the considered HRIR, r is the radius of the sphere and a is the azimuth position of the considered ear (90° for the right ear and 270° for the left one).

A comparison between the original HRIR and its time-shift removed version can be seen in Fig. 5 for the horizontal plane, covering all azimuth angles.

From Fig. 5 one may observe that the initial energy is concentrated at approximately the same time, except for contralateral sound source locations (about 270°), where there is not a well-defined starting energy point, due to the hear barrier and torso diffraction effects.



(a)

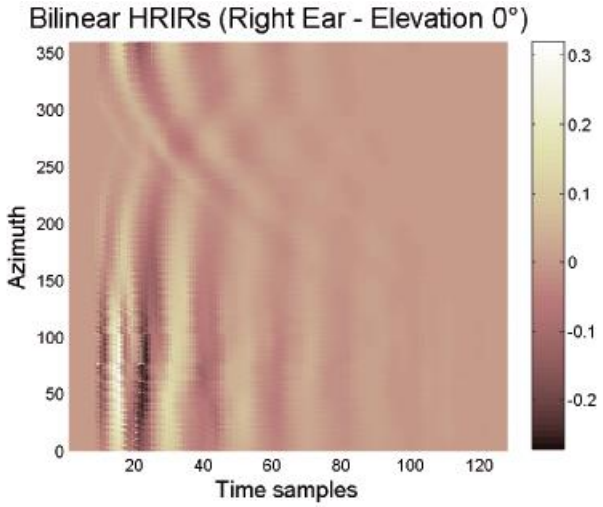


Figure 5: Comparison of delay removal for a fixed elevation in $\theta = 0°$ and azimuth variations of 1° from 0° to 360°: (a) Original HRIRs (b) Delayless HRIR versions.

A. *Overall averaged error*

As mentioned, the MSE is not suitable for evaluating the interpolation quality. The HRIR energy varies according to the sound source location, which may present lower errors for functions with lower energy levels. In this case, the mismatch between target and output can be large, but the MSE may be smaller than when those functions have higher energy levels and smaller differences.

The performance of each network (N_p) can be measured by averaging the MAE for representative directions inside the operating area.

$$N_p = \frac{1}{S}\sum_{s=1}^{S} \mathrm{MAE}_s, \qquad (4)$$

where S is the total number of considered directions, s is a sequential index which represents an individual direction. The considered functions for this calculation had fixed an elevation (0°) and azimuth steps of one degree ($\Delta\phi = 1°$), i.e. the HRIRs computed in the horizontal plane in which the human resolution is more accurate. With this, the optimum number of neurons in the hidden layer can be determined by increasing that number from 2 to 10, for both cases: with and without delays. The results of simulations are shown in Fig. 6, where the overall average errors MAE is presented and compared for several aperture angles and targets.

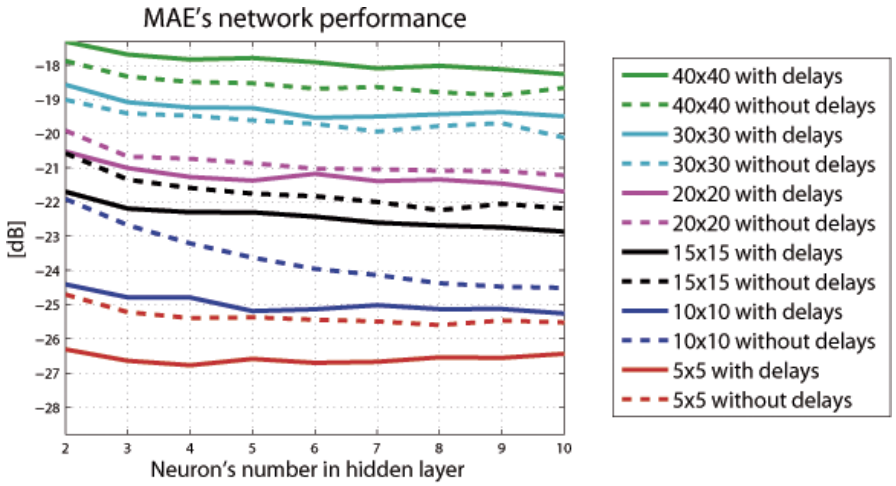


Figure 6: MAE overall average results for different reception areas

Figure 6 presents the MAE for two situations: with original initial delays and when delays were removed. Solid lines represent simulation with original delays. From this figure, one observes that for both metrics the lower errors were achieved by networks with smaller areas. Increasing the areas the error also increases. The smallest error is obtained for targets with original delays. When delays were removed from targets of the networks with the smallest areas (dashed lines) the error becomes 2-3 dB higher. On the other hand, as the aperture size increase, the difference between the errors with and without delays decreases. The approximate size when the errors are similar occurs for networks between $20° \times 20°$ and $30° \times 30°$.

From this error behavior, one may conclude that if the training is performed with smaller areas, the original delays should be preserved, since they contribute with slight time differences in highly correlated targets. For larger areas these delays does not contribute to the learning process. The functions already present variations and lower correlation than when small areas are used. Therefore, by removing such delays, the error for larger areas decreases.

Another conclusion that can be extracted from Fig. 6 is related to the number of neurons in the hidden layer. It is observed that the best performance is found when L=4 is used,

for a 5° × 5° working area and preserving the original delays of the HRIRs. The overall behavior of networks with 5° × 5° with original delays is almost flat and presented variation smaller than 1 dB. Therefore, for such small areas the number of neurons in the hidden layer has a very slight influence in the global behavior.

Errors at higher frequencies may increase objective error metrics, but will not present significant disturbance in the perception of the sound source location [2]. The same occurs for very low frequencies, as stated by Fletcher and Munson in the Equal-loudness contours curves [20] and the ISO 226 standard [21].

V. NUMERICAL RESULTS

In order to present few examples of the proposed interpolation scheme, focusing on specific directions instead of averages of several functions and errors, four measured HRIRs were chosen. From Fig. 7 to Fig. 10, it is presented in the five plots:

a) time domain comparison between output and target;
b) magnitude and (d) phase frequency response comparison;
c) absolute errors in time (upper plot) and magnitude (lower plot).

Such directions – elevation $\theta = 0°$ and azimuths 0°, 90°, 180° and 270° – were chosen due to the their strong variation in spectra, energy and initial delays.

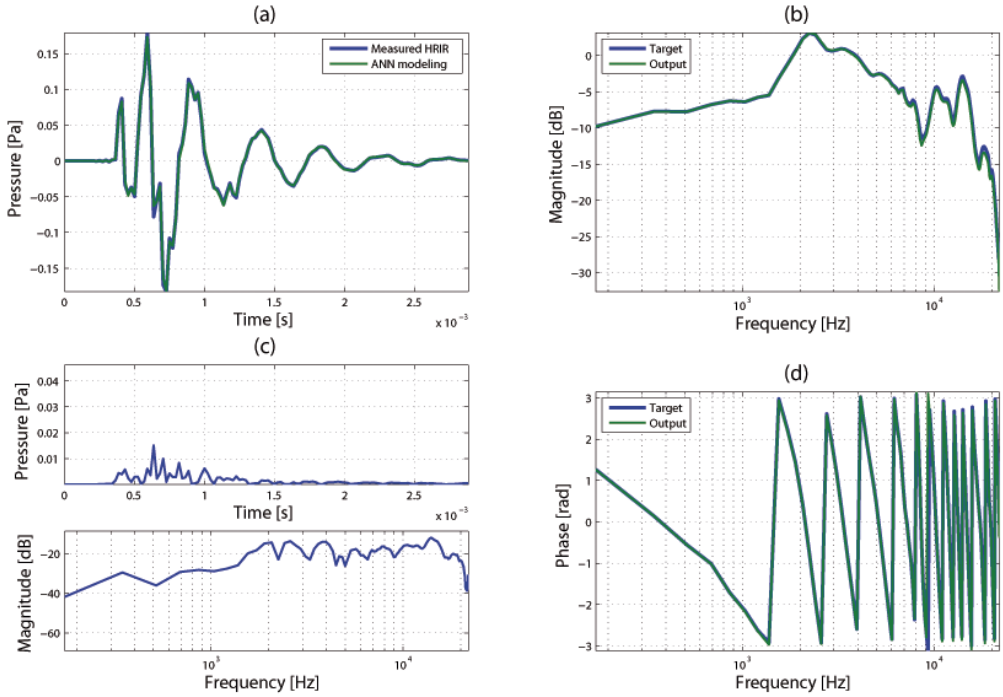


Figure 7: Comparative results between ANN modeling and HRIR measurement: (a) Time domain. (b) Magnitude in the frequency domain. (c) Absolute error in time and frequency domains. (d) Phase in the frequency domain phase

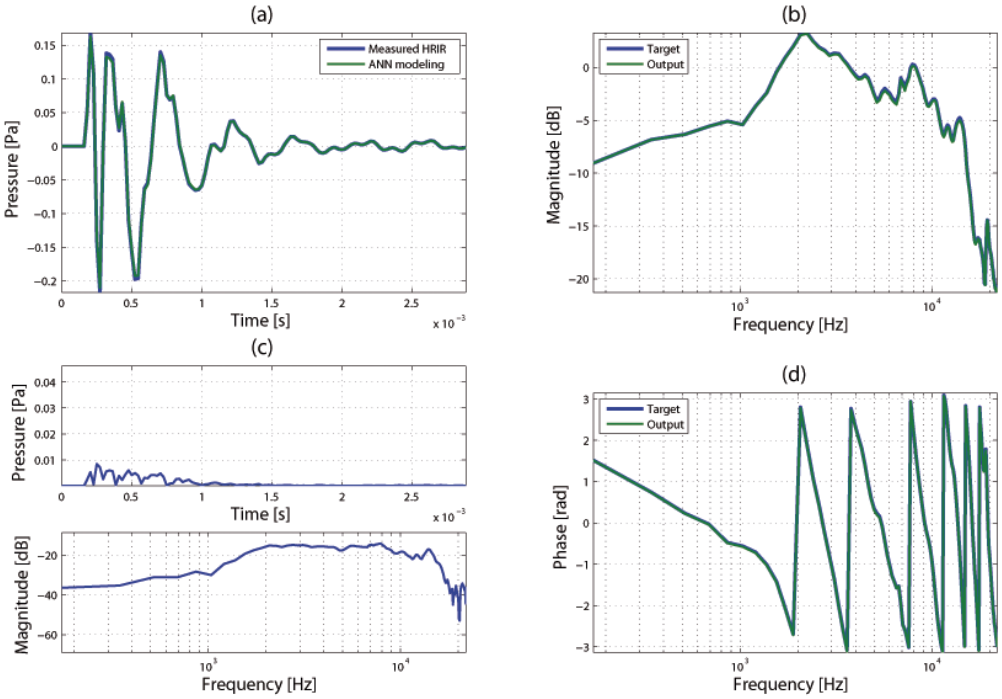


Figure 8: Comparative results between ANN modeling and HRIR measurement: (a) Time domain. (b) Magnitude in the frequency domain. (c) Absolute error in time and frequency domains. (d) Phase in the frequency domain phase

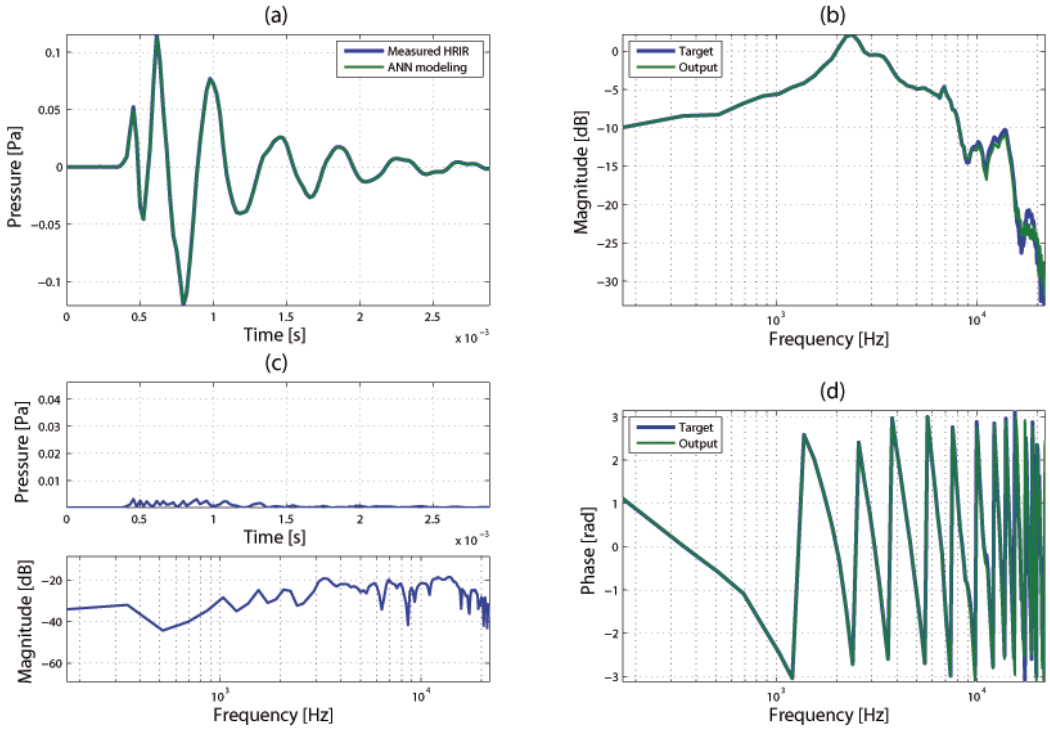


Figure 9: Comparative results between ANN modeling and HRIR measurement: (a) Time domain. (b) Magnitude in the frequency domain. (c) Absolute error in time and frequency domains. (d) Phase in the frequency domain phase

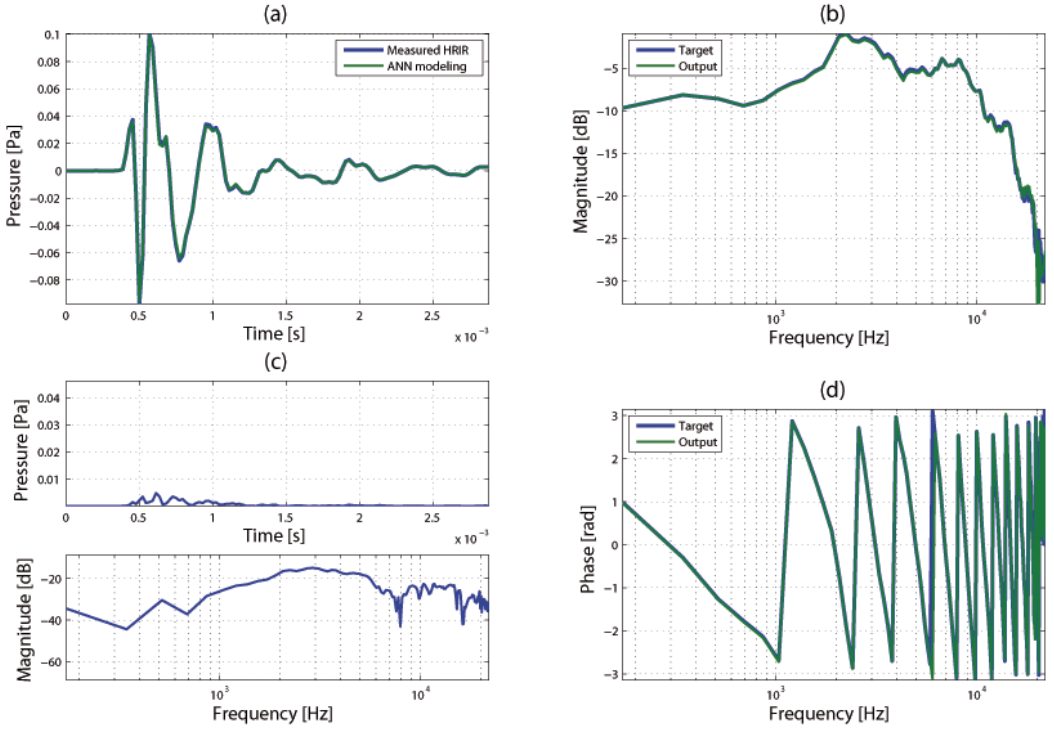


Figure 10: Comparative results between ANN modeling and HRIR measurement: (a) Time domain. (b) Magnitude in the frequency domain. (c) Absolute error in time and frequency domains. (d) Phase in the frequency domain phase

A general comparison between the bilinear technique and the ANN model, for fixed elevation in $\theta = 0°$ and azimuth variations of 1°, are presented in the time domain in Fig. 11 and in the frequency domain (magnitude) in Fig. 12. No noticeable differences are observed.

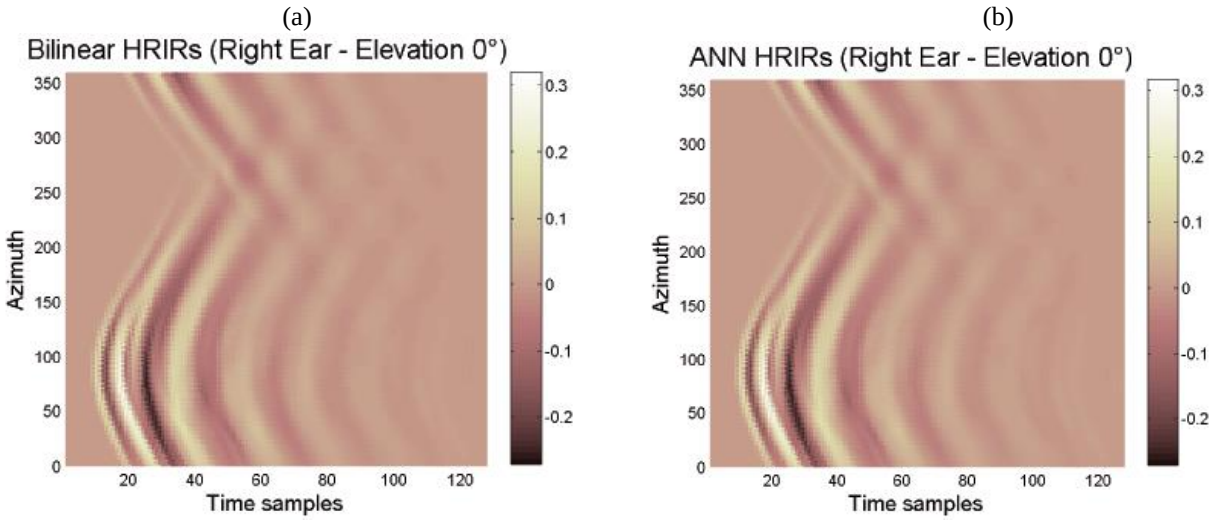


Figure 11: Time domain interpolation results with time shifts for fixed elevation in $\theta = 0°$ and azimuth variations of 1°: (a) Bilinear (b) ANN.

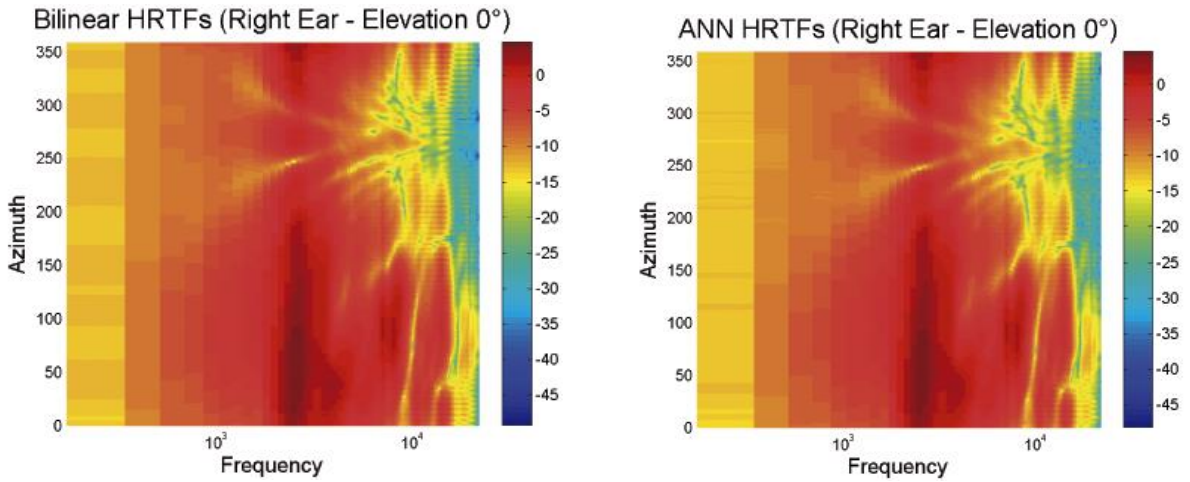


Figure 12: Frequency domain interpolation results with time shifts for fixed elevation in θ = 0° and azimuth variations of 1°: (a) Bilinear (b) ANN.

VI. COMPUTATIONAL COST

This section presents a comparison between the implementation costs of the typical bilinear interpolation method (BIM) and the ANN interpolation scheme presented in this work. The established parameter for such comparison is the required number of elementary arithmetic operations for each method. The ANN execution does not consider the training computational load, since there is no synaptical weights actualization during this phase. Therefore, the number of arithmetic operations A_0 is given by

$$A_0 = 2\left[e \cdot n_1 + \left(\sum_{j=1}^{m-1} n_j \cdot n_{j+1}\right) + n_m \cdot s\right], \qquad (5)$$

where e is the input vector size, n_j is number of neurons in the jth intermediate layer, m is the number of intermediate layers and s is the number of neurons in the output layer.

The results presented in the previous section came from an ANN set with two elements input vector, one intermediate layer with 2 neurons and a 128 neurons output layer. The computational load for the BIM will cost $4 + (6 \times L)$ multiplications and $6 + (3 \times L)$ additions. For L=128, the values indicated in the left column of Table 1 were obtained while the figures corresponding to the ANN, computed in Eq. 5 with $s = 128$, m = 1, $e = 2$ and three differents n_m (2, 3, 4) are presented in the third, fourth and fifth columns of Table 1. By comparing these results it can be stated that the artificial neural network model presents a computational complexity reduction of almost 49.83%.

VII. APPROACHES FOR OBSTACLE DETECTION

Once the technique for interpolate HRIRs had proven its accuracy and computational convenience, the next step is to generate a 3D sound emitter. This can be achieved by implementing a convolution product between the ANN's result and an arbitrary anechoic sound. This procedure can be found in the broadly majority of signal processing signals books.

Nevertheless, in order to have means to apply this solution, the ANNs inside the 3D sound emitter must be feed with spherical coordinates that pin point the obstacle position. Therefore, a device with obstacle detection capabilities must be provided. Two approaches will be developed for this end.

A. Computational Vision for obstacle detection and proximity

Artificial vision is one of artificial intelligence techniques applied for many years for the detection of different types of objects. Therefore, it is proposed to develop an artificial vision system for the detection and classification of obstacles. In this sense, several techniques could be applied. For instance, Stereo Vision [22] allows the identification of objects and proximity in three-dimensional space. This information can be transformed in distance and position of the detected obstacles. In a second phase, an ANN could be used for classification and interpretation of relevant obstacles characteristics. Figure 13 shows how obstacles can be represented in real time through computer vision.

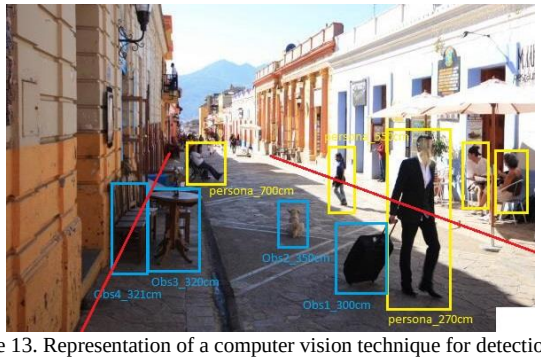


Figure 13. Representation of a computer vision technique for detection and classification of objects.

TABLE 1
COMPARISON OF THE COMPUTATIONAL COST BETWEEN BIM AND ANN INTERPOLATION OF FUNCTION WITH L = 128. THREE ARCHITECTURES ARE CONSIDERED.

Detail	BIM	n_1=2	n_1=3	n_1=4
Number of additions (+)	390	260	390	520
Number of multiplication (×)	772	260	390	520
Computational gain (+)	-	33.33%	0.00%	-33.33%
Computational gain (×)	-	66.32%	49.48%	32.64%
Mean computational gain	-	49.83%	24.74%	-0.35%

The phases for obstacle detection and classification are described in Fig. 14.

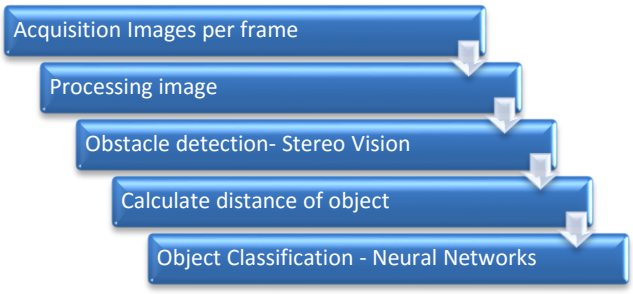


Figure 14. Phases for detection and classification of objects.

B. *Laser scanning for obstacle detection and proximity measurement*

Modern electronic sensor systems for obstacle detection are widely used in robotics applications and assistance to people.

One of the most effective approaches to apply this relies on laser proximity sensors. Such approach exhibit excellent benefits for obstacle detection and proximity measurement without contact or friction with the objects in question. Distance measurement by these devices is done in the following ways:

- Flight time [23]
- Phase shift

Based on the aforementioned methods it is essential that the sensor has the ability to scan the scene taking *n* samples as shown in Fig. 15.

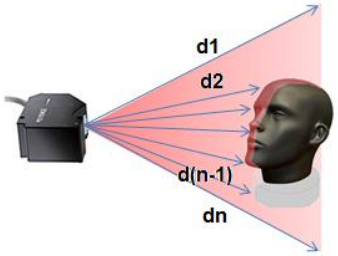


Figure 15. Laser Scanning [24]

Depending on the opening angle of the laser sensor it is possible to detect a point cloud to identify the presence of objects. Adding a sensor that has the ability to perform a vertical scan can have more information and a more detailed point cloud.

The point cloud provides a wealth of information to the user, allowing to pin point the object's position and distance relative to user that acts as a reference frame.

The steps to identify the obstacles through the use of this type of sensor are shown in Fig. 16.

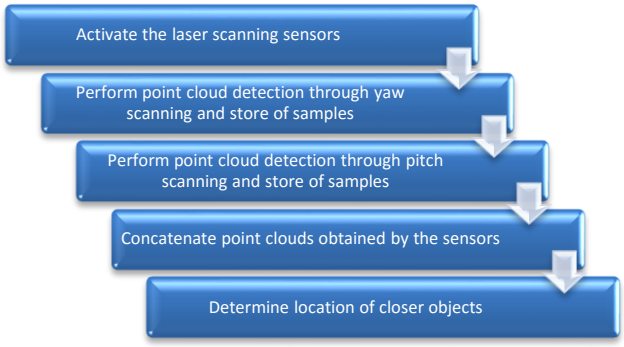


Figure 16. Phases for laser scanning for obstacle detection and proximity measurement

VIII. CONCLUSIONS

The main goal of this study was to present an interpolation procedure for HRIRs by using a committee of artificial neural networks. The Mean Absolute Error (MAE) results show that the obtained accuracy refrains the use of reception areas with smaller angle apertures. The time shifts, although more subtle in smaller reception areas than those found in bigger ones, still constitutes a learning pattern for networks with the adequate capacity to learn from it. Such capacity is defined by the number of neurons in the hidden layer.

The model performance presents larger deviations in high frequency components which are faded away because of the generalization property in the ANN modeling. Nevertheless, these deviations can be neglected due to the low energy present in such frequencies and also to the limited characteristics of the human hearing in high frequencies.

In summary, the size of the reception area is a critical element to be considered for ensuring a high precision interpolator. A classical interpolation technique, such as a bilinear one, could be substituted by an ANN whose output presented very small errors if compared with the corresponding target functions.

Preliminary comparisons performed in time and frequency domains with actual measured functions, show that an ANN committee of reduced architecture (with 1 hidden layer with at most 2-4 neurons), working inside $5° \times 5°$ reception areas and trained with functions with original delays, is able to substitute an interpolation method with a computational reduction of almost 50% while keeping a similar precision.

Of course, by using such small reception areas, the system must deal with a considerable increment of the number of networks to be used and involves a larger spent in memory resources. Nevertheless, current computers have enough memory to work with such memory requirements without compromising the result's generation speed.

The next step of this research is to implement a convolution product process in order to insert the 3D sound effect of the interpolated HRIR into an arbitrary signal (3D sound emitter). At the same time, the obstacle detection approaches will be implemented in proper hardware architecture in order to feed spatial coordinates to the 3D sound emitter.

REFERENCES

[1] M. Vorländer, 2008. Auralization: Fundamentals of Acoustics, Modelling, Simulation, Algorithms and Acoustic Virtual Reality. Springer, Berlin, pp. 86–87

[2] J. Blauert, 1997. Spatial Hearing. The MIT Press, Cambridge, pp. 372–392.

[3] [3] G. Wersnyi, 2009. Effect of emulated head-tracking for reducing localization errors in virtual audio simulation. IEEE Transactions On Audio, Speech, And Language Processing, vol. 17, n. 2, pp. 247–252.

[4] T. Ajdler, C. Faller , L. Sbaiz and M. Vetterli, 2005. Sound Field Analysis along a Circle and Its Applications to HRTF Interpolation. The Journal of the Audio Engineering Society, vol. 56, n. 3, pp. 156–175.

[5] H. Hacihabiboglu, B. Gunel and A. M. Kondoz, 2005. Head-related transfer function filter interpolation by root displacement. In Proc. IEEEWorkshop on Applications of Signal Processing to Audio and Acoustics (WASPAA-05), New Paltz, NY, USA, pp. 134–137.

[6] T. Nishino, N. Inoue, K. Takeda and F. Itakura, 2007. Estimation of HRTFs on the horizontal plane using physical features. Applied Acoustics, vol. 68, pp. 897–908.

[7] N. H. Adams and G. H. Wakefield, 2008. State-space synthesis of virtual auditory space. IEEE Transactions On Audio, Speech, And Language Processing, vol. 16, n. 5, pp. 881–890.

[8] G. Enzner, 2009. 3D-continuous-azimuth acquisition of head-related impulse responses using multi-channel adaptive filtering. In Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA-2009), New Paltz, NY, USA, pp. 325–328.

[9] M. Pollow and M. Vorl¨ander, 2011. Deriving continuous HRTFs from discrete data points. Fortschritte der Akustik: DAGA 2011; 37. Jahrestagung f¨ur Akustik; 21. Dsseldorf / DEGA.Wiss. Ed. J. Becker-Schweitzer. pp. 641–642.

[10] J. Ahrens, M. R. P. Thomas and I. Tashev, 2012. HRTF magnitude modeling using a non-regularized least-squares fit of spherical harmonics coefficients on incomplete data. In Proc. Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC), Asia-Pacific, Dec. 2012, pp. 1–5.

[11] J. F. Lucio Naranjo, R. A. Tenenbaum and J. C. B. Torres, 2010. Using Artificial Neural Networks to generate virtual acoustic reality applied on escape training in blind conditions. International Review of Chemical Engineering (I.RE.CH.E.), Vol. 2, No. 6, pp. 754–759.

[12] Y. Liu and B. Xie, 2012. Analysis on the stability of spatial interpolation schemes for head-related transfer function. The Journal of the Acoustical Society of America, vol. 131, n. 4, pp. 3305–3305.

[13] J. Tebelskis, 1995. Speech Recognition using Neural Networks. Ph.D. Thesis - Carnegie Mellon University, Pittsburgh, pp. 4–7.

[14] D. Rumelhart, J. McClelland and the PDP Research Group, 1986. Parallel Distributed Processing: Explorations in the Microstructure of Cognition. MIT Press, pp. 328–330.

[15] S. Haykin, 2009. Neural Networks and Learning Machines. Prentice Hall, New Jersey, pp. 21–24.

[16] B. Gardner, K. Martin, 1995. HRTF Measurements of a KEMAR Dummy-Head Microphone. The Journal of the Acoustical Society of America, vol. 97, n. 6, pp. 3907–3908.

[17] J. C. B. Torres, M. R. Petraglia and R. A. Tenenbaum, 2004. An Efficient wavelet-based HRTF for auralization. Acta Acustica united with Acustica, vol. 90, n. 1, pp. 108–120.

[18] Z. Haraszy, D. Ianchis and V. Tiponut, 2009 Generation of the Head Related Transfer Functions Using Artificial Neural Networks. 13th WSEAS International Conference on Circuits, ISBN: 978-960-474-096-3, ISSN: 1790-5117, pp. 114–118.

[19] E. A. Macpherson and J. C. Middlebrooks, 2002 Listener weighting of cues for lateral angle: The duplex theory of sound localization revisited. The Journal of the Acoustical Society of America, vol. 111, n. 5, pp. 2219–2236.

[20] H. Fletcher and W. A. Munson, 1933. Loudness, its definition, measurement and calculation. The Journal of the Acoustical Society of America, vol. 5, pp. 82–108.

[21] ISO226, 2003. Acoustics – Normal equal-loudness-level contours.

[22] D. Marr and T. Poggio, 1979. A computational theory of human stereo vision. Proceedings of the Royal Society of London B: Biological Sciences, 204(1156), 301-328.

[23] J. Corso, 2012. Escáner de Tiempo de vuelo y de triangulación. [Online]. Available: http://www.surveyterra.com/2012/06/escaner-de-tiempo-de-vuelo-y-de.html

[24] L. Ilbay. “Reconstrucción activa de objetos 3d mediante escaneo láser y generación de la vista por medio del software MatLab”. Engineering Monography. Dept. Control and Automatization. Eng., National Polytechnic School, Quito, 2014.

José Francisco Lucio Naranjo was born in Quito – Ecuador, in June 1st of 1979. He has an engineer degree in Computer Systems by the Pontificia Universidad Católica del Ecuador (2005). In 2010 and in 2014, he obtained, respectively, a M.Sc. and a Ph.D degrees, both in Computer Modeling by IPRJ, Rio de Janeiro State University, Brazil. His major field of study is computational modeling focus on acoustic numeric simulation applied in virtual reality using artificial neural networks.

Dr. Lucio is currently acting as computer science titular professor at the National Polytechnic School (EPN) of Ecuador. He has also acted as substitute teacher at the Rio de Janeiro State University (UERJ), distance education tutor in the Federal Fluminense University (UFF), curricular professor at the Universidad de las Américas (UDLA) and auxiliary professor at the Central University of Ecuador (UCE). He was also an associate member of the Acoustical Society of Brazil (SOBRAC). (UCE). También ha sido miembro asociado de la Sociedad Brasileña de Acústica (SOBRAC).

Roberto A. Tenenbaum is a Mechanical Engineer by the Federal University of Rio de Janeiro, in 1972; obtained his M.Sc. in the field of Thermoelasticity by COPPE, Federal University of Rio de Janeiro, in 1975 and his D.Sc. in the field of Acoustics by the same university, in 1987. His major field of interest is dynamics, acoustics and vibration.

He has worked as Engineer at the Nuclear Technology Company, in Rio de Janeiro, from 1973 to 1974. Then, as a Professor and Researcher at the Acoustics and Vibration Lab. (LAVI) in the Graduate Program for Mechanical Engineering, in the Federal University of Rio de Janeiro, in Rio de Janeiro, Brazil, from 1974 to 2004. Now, he works as a Professor and Researcher at the Dynamics, Acoustics and Vibration Lab. (LIDAV) in the Graduate Program in Computer Modeling, in the State of Rio de Janeiro University, in Nova Friburgo, Brazil. He is the author of three books on dynamics: Dinâmica, 1997, in Portuguese; Fundamentals of Applied Dynamics, 2004, in English, published by Springer-NY; and Dinâmica Aplicada, 2006, also in Portuguese. He is also the author or co-author of more than 150 articles published in journals and conferences.
Professor Tenenbaum is an honorary and founder member of the Brazilian Society of Mechanical Sciences and Engineering, ABCM; a founder member of the Acoustical Society of Brazil, SOBRAC; is member of The Acoustical Society of America, ASA; and the International Society for Inverse Problems in Science and Engineering, ISIPSE, among others.

Luis Alberto Morales Escobar was born in Quito – Ecuador, in January 14th of 1985. He has an engineer degree in Electronics by the Escuela Politécnica Nacional (2010). In 2012, he obtained a M.Sc. in Automatics and Robotics, in Universitat Politecnica de Catalunya, Barcelona- Spain. His major field of study is Computer Vision and Robotic Systems. MSc. Luis Morales is currently acting as titular professor at the Escuela Politécnica Nacional of Ecuador, and he is manager of Electronic Instrumentation Laboratory of the Faculty of Electrical and Electronic Engineering from the EPN.

Henry Patricio Paz Arias was born in Zamora – Ecuador, in Abril 14st of 1986. He has an engineer degree in Systems by the Universidad Nacional de Loja Ecuador (2010). In 2012 he obtained the Master in Computer Science in the Area of Artificial Intelligence. Currently he is pursuing a PhD in computer science in the area of intelligent systems of the National Polytechnic School (EPN) of Ecuador. Ing. Paz is currently acting as computer science titular professor at the National Polytechnic School (EPN) of Ecuador. He has also acted as teacher in the Universidad Interglogal - Pachuca - México, National University of Loja - Ecuador teaching materials artificial intelligence.

Carlos Efraín Iñiguez Jarrín was born in Quito – Ecuador, in November 15th of 1979. He got an engineer degree in Systems Computer at Escuela Politécnica Nacional (2005). In 2013, he obtained a M.Sc. in Web Engineering, at Universidad Politécnica de Madrid, Spain. His major field of study is Software Engineering. MSc. Carlos Iñiguez is acting as titular professor at Escuela Politécnica Nacional of Ecuador (EPN), and he is currently leading the software development for the research project ¨Simulation of Acoustic Wave Propagation in closed areas¨, at EPN.

Published by:

National Polytechnic School
Faculty of Systems Engineering
Department of Informatics and Computer Sciences
Ecuador

http://lajc.epn.edu.ec/
lajc@epn.edu.ec

November 2015