INSTITUCIÓN UNIVERSITARIA
SALAZAR Y HERRERA
Una decisión que se nota.

Quid, N°. 18, pp. 25-34, Ene-Jun, 2012, ISSN: 1692-343X, Medellín-Colombia

IMPLEMENTACIÓN DE LA ARQUITECTURA PELEA POR MEDIO DE SERVICIOS WEB SEMÁNTICOS PARA LA MANIPULACIÓN REMOTA DE ROBOTS

PELEA ARCHITECTURE IMPLEMENTATION THROUGH SEMANTIC WEB SERVICES FOR A REMOTE ROBOTS MANIPULATION

PhD. Jaime Alberto Guzmán-Luna
Universidad Nacional de Colombia, *Facultad de Minas, Ingeniería de Sistemas e Informática, SINTELWEB. Carrera 80 # 65-223, Medellín, Colombia jaguzman@unal.edu.co*

Est. Juan David Meza González
Universidad Nacional de Colombia, *Facultad de Minas, Ingeniería de Sistemas e Informática, SINTELWEB. Carrera 80 # 65-223, Medellín, Colombia judmezago@unal.edu.co*

Est. Paola Andrea Galeano Hincapié
Universidad Nacional de Colombia, *Facultad de Minas, Ingeniería de Sistemas e Informática, SINTELWEB. Carrera 80 # 65-223, Medellín, Colombia pagaleanoh@unal.edu.co*

(Recibido el 07-09-2011. Aprobado el 01-02-2012)

Resumen: este artículo presenta la implementación de un sistema para la manipulación de robots, basado en la arquitectura PELEA haciendo uso de servicios web para la manipulación remota de robots para simular la construcción de diques. Dichos servicios fueron descritos semánticamente por medio de la ontología OWL-S para facilitar su invocación tanto por máquinas como humanos proporcionando así un sistema capaz de llevar a cabo tareas relacionadas con la construcción de un dique. Para esto se toma como guía el reglamento establecido para competencias internacionales sobre construcción de diques por medio de robots (UDI, 2012).

Palabras clave: robótica; robótica en la nube; servicios web; web semántica.

Abstract: *this paper presents the implementation of a system for handling robots, based on the PELEA architecture using web services for remote robot manipulation to simulate the dam constructions. These services were described semantically using OWL-S ontology to facilitate invocation both machines and humans providing a system capable of carrying out tasks related to the construction of a dam, for this is taken as a guide to the rules established international competitions dam construction by robots (UDI, 2012).*

Keywords: *robotics, cloud robotics, web services, semantic web.*

1. INTRODUCCIÓN

Hoy se advierte una tendencia mundial para conseguir que los robots, sin importar el tipo, la plataforma o el sistema operativo que usen, estén en capacidad de comunicarse entre sí por medio de mensajes remotos intermediados por servidores (Lafuente y Larrea, 2009), que estarían ejecutando servicios web (Cavanaugh, 2006) y que permitan desarrollar cualquier tarea propuesta de forma remota.

Uno de los principales aportes del manejo remoto de un robot es que, como existe un servidor o servidores remotos que intervienen en el proceso, es posible disminuir la carga computacional que el robot debe soportar (Hestand, 2011), es decir, ya no sería el robot el encargado de los cálculos, análisis y ejecución de algoritmos, sino que parte de estas tareas serían asignadas a los servidores remotos como parte integrada del sistema robótico. Ello facilita el aumento de la capacidad de cómputo del sistema asociado con el robot para incrementar la complejidad de los procesos y la capacidad de procesamiento por parte de los servidores asociados.

Esto genera un nuevo tipo de robot, más económico, capaz de realizar innumerables tareas y para diferentes propósitos, haciéndolos más accesibles para el uso personal (WillowGarage, 2011). Este artículo se enfoca en mostrar la implementación de un sistema para la manipulación de robots, basado en la arquitectura PELEA "ya que esta arquitectura posee componentes y/o módulos, capaces de realizar la planificación, ejecución, seguimiento y aprendizaje, de una manera integrada, por lo que resulta adecuada para una amplia gama de problemas de planificación" (Alcázar et al., 2010, p. 1), y que por consiguiente se puede aplicar a este caso de estudio, con el fin de integrarla a servicios web semánticos (Vaculin y Svcara, 2007), donde cada módulo que compone a esta arquitectura está implementado como un servicio web que puede ser invocado de manera independiente por medio de su descripción semántica asociada. Dichos servicios fueron descritos semánticamente con el lenguaje OWL-S (Martin et al., 2004) sobre una ontología de dominio (Bovce y Pahl, 2007) que proporciona un dominio y un problema PDDL (Gala et al., 2003) que son usados posteriormente para la planificación de las acciones que el robot deberá llevar a cabo. De este modo, usando la descripción semántica de estos servicios, se hace posible su invocación con el uso de la API de OWL-S (Siren y Parsia, 2004).

Este documento está organizado de la siguiente manera: en la sección II se detalla la arquitectura propuesta para el sistema y se muestran las capas que la conforman. En la sección III se muestra el escenario y las características de éste para simular el problema planteado de construir un dique. En la sección IV se explican el dominio y el problema PDDL usados por el módulo de planificación de la arquitectura para solucionar el problema de construir un dique y se muestra su integración con una ontología para darles un sentido semántico al interior del sistema. Esta ontología es usada para generar desde el módulo de planificación un plan de las acciones que el robot deberá llevar a cabo y así solucionar el problema propuesto. En la sección V se detalla la implementación de los principales módulos de la arquitectura PELEA por medio de los servicios web, enfocados hacia la construcción de un dique y se muestra el programa que controla de manera remota al robot. En la sección VI se hace explícita la forma en la que se realizó la descripción semántica de los servicios web. La sección VII presenta cómo se efectúa la invocación del sistema desde el módulo ejecutor a partir de su descripción semántica enviando como parámetros un problema y un dominio PDDL obtenidos desde la ontología. Finalmente, en la sección VIII se presentan las conclusiones y trabajos futuros de este proyecto.

2. METODOLOGÍA PROPUESTA

Para soportar este trabajo se propone una metodología compuesta por los siguientes pasos: la implementación física del robot, la implementación de los servicios web que representan cada módulo de la arquitectura PELEA (planificación, traducción y ejecución) y que controlan remotamente el robot, la creación del marcado semántico de dichos servicios y la implementación del módulo para la invocación estos servicios web a partir de su descripción semántica. (Ver Fig.1). A lo largo de este artículo se detallarán los pasos de la metodología seguida.

Fig. 1. Metodología propuesta

Como resultado de seguir esta metodología se obtuvo un sistema robótico manejado remotamente en el que el robot se comunica con un servicio web, que constituye un módulo de la arquitectura PELEA y a su vez posee una descripción semántica que hace uso de una ontología de dominio (ver figura 5) en asocio con una ontología para la descripción de servicios (ver figura 12) y finalmente, haciendo uso de las anteriores ontologías, se puede invocar cada servicio por medio de una aplicación orientada al manejo de descripciones semánticas de servicios web (ver figura 13).

3. CREACIÓN DE COMPONENTES PARA LA IMPLEMENTACIÓN DE LA ARQUITECTURA PELEA USANDO SERVICIOS WEB.

3.1. Descripción del escenario planteado para simular el problema de construir un dique.

Como problema para este sistema, se propuso un escenario similar al usado en competencias robóticas como la LARC (IEEE, 2011) en la categoría de constructor de diques, a continuación se describe el escenario:

C100	C101	C102	C103	C104	C105	C106	C107	C108	C109	C110
C89	C90	C91	C92	C93	C94	C95	C96	C97	C98	C99
C78	C79	C80	C81	C82	C83	C84	C85	C86	C87	C88
C67	C68	C69	C70	C71	C72	C73	C74	C75	C76	C77
C56	C57	C58	C59	C60	C61	C62	C63	C64	C65	C66
C45	C46	C47	C48	C49	C50	C51	C52	C53	C54	C55
C34	C35	C36	C37	C38	C39	C40	C41	C42	C43	C44
C23	C24	C25	C26	C27	C28	C29	C30	C31	C32	C33
C12	C13	C14	C15	C16	C17	C18	C19	C20	C21	C22
C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11
1	2	3	4	5	6	7	8	9	10	11

DESBORDAMIENTO SIMPLE

DONDE SE DEBEN UBICAR LOS PILOTES

DESBORDAMIENTO

UBICACIÓN DE LOS PILOTES

POSICIÓN DEL ROBOT

Fig. 2. Escenario utilizado para probar el sistema

```
(define (domain Cuadriculas)
    (:requirements :typing :fluents)
    (:types Celda - object
            Objetivo - object
            Robot - object
    )
    (:predicates
        (Vacia ?x - Celda)
        (ContiguaN ?x - Celda ?y - Celda)
        (ContiguaE ?x - Celda ?y - Celda)
        (ContiguaW ?x - Celda ?y - Celda)
        (ContiguaS ?x - Celda ?y - Celda)
        (ObjetivoEn ?x - Objetivo ?y - Celda)
        (Tiene ?x - Objetivo ?y - Robot)
        (Disponible ?x - Robot)
        (Disp ?x - Objetivo)
        (EnCelda ?x - Celda ?Yo - Robot)
    )
    (:action ViajarN
        :parameters   (?Origen - Celda ?Destino
        :precondition (and (EnCelda ?Origen ?Yo
        :effect (and (EnCelda ?Destino ?Yo) (no
                )
    )
    (:action AgarrarN
        :parameters (?Target - Objetivo ?Ubicac
        :precondition (and (EnCelda ?UbicacionR
        :effect (and (not(ObjetivoEn ?Target ?U
                )
    )
    (:action SoltarN
        :parameters (?Target - Objetivo ?Origen
        :precondition (and (Tiene ?Target ?Yo)
        :effect (and(not(Tiene ?Target ?Yo))(Di
                )
```

Fig. 3. Segmentos del dominio PDDL

Tal como se aprecia en la Fig. 2, el escenario usado consta de 110 celdas, una columna central que representa el río y una segunda columna que representa el desborde de dicho río, por lo tanto el objetivo del sistema es conseguir que el robot coloque los pilotes ubicados en el extremo como posición inicial y ponerlos en la posición final alrededor del desborde. Para este problema se consideró un desborde simple localizado en el extremo derecho del escenario y se tomó como posición inicial del robot la celda 98.

3.2 Dominio y problema PDDL usados desde una ontología.

Uno de los principales módulos de la arquitectura PELEA es el módulo de planificación, el cual recibe un dominio y un problema de alto nivel y genera el plan con las acciones que el robot deberá ejecutar para llevar a cabo el objetivo propuesto.

Dado que se pretende tener un sentido semántico para todos los componentes del sistema, tanto el dominio como el problema PDDL, se encuentran alojados en una ontología accesible desde una dirección remota (Ceravolo et al., 2006). A continuación se especifican segmentos del dominio y el problema usados para la simulación de un constructor de diques y la representación de estos al interior de la ontología.

Como se aprecia en la Fig. 3, se definieron las celdas que el robot deberá recorrer, los objetivos que éste deberá manipular y el robot como tal. Adicionalmente, se definen las celdas que son contiguas por el norte, sur, este y oeste; cuáles de las celdas están vacías; la localización de los objetivos al interior del escenario; la disponibilidad de un objetivo y la localización actual del robot en el escenario; se definen acciones que el robot puede ejecutar como son: viajar al norte, tomar un pilote ubicado al oeste o soltar un pilote en el norte. Se debe advertir que se definieron operaciones para viajar, agarrar y soltar en los cuatro puntos cardinales.

```
(define (problem Desbordamiento)
  (:objects
      C1  -Celda
      C8  -Celda
      C9  -Celda
      C10 -Celda
      C109    -Celda
      C110    -Celda
      Lego - Robot
      T11 - Objetivo
      T12 - Objetivo
  )
  (:init
  Vacia   C4  )
  Vacia   C5  )
  Vacia   C107    )


      (   ContiguaN   C1  C12 )
  ContiguaN   C2  C13 )
  ContiguaE   C1  C2  )
  ContiguaW   C110    C109    )
  ContiguaS   C12 C1  )

      (Disponible Lego)
      (EnCelda C98 Lego)
      (Disp T1)
      (Disp T12)
      (ObjetivoEn T1 C108)
      (ObjetivoEn T2 C109)
      (:goal
          (and
              (ObjetivoEn T1 C95)
              (ObjetivoEn T2 C84)
              (ObjetivoEn T8 C30)
              (ObjetivoEn T9 C19)
              (ObjetivoEn T10 C18)
              (ObjetivoEn T11 C7)
              (ObjetivoEn T12 C9)

          )
      )
```

Fig. 4. Segmentos del problema PDDL

Como se aprecia en la Fig. 4, se describieron las 110 celdas, luego se expresó cuáles eran contiguas por el norte, el sur, el este y el oeste; se mostró cuales casillas se encuentran vacías, cuales objetivos estaban disponibles; la ubicación del robot, la ubicación de los objetivos y la meta a alcanzar según la posición final que debían alcanzar los pilotes.

A continuación se muestra ontología que permite insertar, como instancias de éste, el problema y el dominio PDDL anteriormente descritos.

```
<owl:Class rdf:ID="PlanH"/>
<owl:Class rdf:ID="DomainL"/>
<owl:Class rdf:ID="StateH"/>
<owl:Class rdf:ID="InfoMonitor"/>
<owl:Class rdf:ID="Problem"/>
<owl:Class rdf:ID="StateL"/>
<owl:Class rdf:ID="DomainH"/>
<owl:Class rdf:ID="PlanL"/>
<owl:Class rdf:ID="FinishProblem"/>
<owl:DatatypeProperty rdf:ID="content">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#DomainH"/>
        <owl:Class rdf:about="#PlanH"/>
        <owl:Class rdf:about="#Problem"/>
                    ⋮
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:range
    rdf:resource=
    "http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
```

Fig. 5. Ontología para representar el dominio y el problema PDDL

Tal como se aprecia en la Fig. 5, se creó una ontología de dominio (Brusa et al., 2006) con nueve clases que corresponden a las diferentes entradas y salidas de los módulos de la arquitectura PELEA, a continuación se muestra la ontología que contiene las instancias específicas del DomainH y Problem, usados como parámetros para el módulo ejecutor del sistema.

```
<owl:Ontology rdf:about="&local;">
  <owl:imports
    rdf:resource="&local;Domain.owl"/>
</owl:Ontology>
<domain:Problem rdf:ID="Problem_1">
  <domain:content
    rdf:datatype="&schema;string">
      (define (problem Desbordamiento)
</domain:Problem>
<domain:DomainL rdf:ID="DomainL_1">
  <domain:content
    rdf:datatype="string">
  </domain:content>
</domain:DomainL>
<domain:DomainH rdf:ID="DomainH_1">
  <domain:content
    rdf:datatype="&schema;#string">
      (define (domain Cuadriculas)
</domain:DomainH>
```

Fig. 6. DomainH y Problem instanciados

3.2 Principales módulos de la arquitectura PELEA implementados con servicios web.

Un servicio web es una aplicación ejecutable en un servidor remoto y es accesible por medio de protocolos de comunicación, independientes de la plataforma o sistema operativo de quien realiza la comunicación e independiente servidor en el que se encuentra alojado el servicio (Gunner, 2002).

Si un servicio web es capaz de establecer una comunicación con un robot y enviarle a éste una acción para ejecutar, es posible conseguir que un robot sea controlado de manera remota. En este punto se puede expresar que los servicios web son un soporte básico de un área actual de investigación llamada robótica en la nube (Hu et al., 2011).

A continuación se exponen los principales módulos de la arquitectura PELEA que se implementaron por medio de servicios web para la manipulación remota del robot:

- **Módulo Ejecutor**

El módulo ejecutor recibe principalmente el DomainH y el Problem, directamente como instancias de la ontología, por esta razón se le hace un tratamiento a estas entradas para obtener directamente los contenidos de estos y enviarlos posteriormente al monitor.

```
public class Executor
{
    public String peleaExecutor(String p, String dH, String dL)
      throws IOException, InterruptedException
    {
        p = tratarString(p);
        dH = tratarString(dH);

        String pL;
        pL =    new Monitor().peleaMonitor(p, dH, dL, "");

        //Se ejecuta el plan desde consola
        String cmd= "nxjpc.bat Prueba_BT ";
        Process p = Runtime.getRuntime().exec(cmd + pL);

        return pL;
    }

    private String tratarString(String cadena)
    {
        int inicial = cadena.indexOf(">");
        int fin = cadena.indexOf("<", inicial);
        cadena = cadena.substring(inicial,fin);
        return cadena;
    }
}
```

Fig. 7. Módulo ejecutor como servicio web

Luego lo recibido por el monitor (PlanL), es enviado por medio de un proceso externo (Linglom, 2007) al robot.

- **Modulo Monitor**

```
public class Monitor
{
   public String peleaMonitor(tring p, String dH, String dL, String sL)
       throws IOException, InterruptedException
   {
       String sH;
       sH = new Low_To_High_Translator().peleaLowToHighTranslator(sL);
       String mP;
       mP= new Goal_Metric_Generation().peleaGoalMetricGeneration(dH, p);
       String fullProblem = sH + mP + p;
       String pH
       pH = new Decision_Support().peleaDecisionSupport(dH,fullProblem);
       String pL
       pL = new Low_Level_Planner().peleaLowLevelPlanner(pH, dL);
       return pH + pL;
   }
}
```

Fig. 8. Módulo monitor como servicio web

El monitor es el módulo central de esta arquitectura, éste recibe como parámetros el DomainH y el Problem para enviarlos a los respectivos módulos que se encargarán de la planificación de las acciones para el robot, y de la traducción de este plan a uno de bajo nivel para ser comprendidas por el robot.

- **Módulo de Planificación**

```
public String peleaHighLevelReplanner(String dH, String p)
    throws IOException, InterruptedException
{
    FileWriter problem
    problem = new FileWriter("D:/LPGPlanner/Problem.pddl");
    FileWriter d
    d = new FileWriter("D:/LPGPlanner/Domain.pddl");
    BufferedWriter out1 = new BufferedWriter(problem);
    BufferedWriter out2 = new BufferedWriter(d);
    out1.write(p);
    out2.write(dH);
    out1.close();
    out2.close();

    String[] command = new String[9];
    command[0] = "cmd";
    command[1] = "/C";
    command[2] = "D:\\LPGPlanner\\lpg-td-1.0.exe";
    command[3] = "-o";
    command[4] = "D:\\LPGPlanner\\Domain.pddl";
    command[5] = "-f";
    command[6] = "D:\\LPGPlanner\\Problem.pddl";
    command[7] = "-speed";
    command[8] = "-noout";

    Process pr = Runtime.getRuntime().exec(command);
```

Fig. 9. Módulo de planificación como servicio web

El módulo de planificación hace uso del planificador LPG (Gerevini y Serina, 2002) para la generación del plan que el robot deberá llevar a cabo. Este módulo hace un llamado al ejecutor externo del LPG que genera el plan y retorna únicamente el contenido de éste, capturando lo mostrado por la ventana de comandos (Linglom, 2007).

- **Módulo de Bajo Nivel**

El módulo de bajo nivel, es el encargado de convertir el plan de alto nivel enviado por el monitor a un conjunto de acciones de bajo nivel que el robot sea capaz de comprender.

```
// 1 = izq, 2 = der, 3 = adlte, 4 = atrs
public static void giroYAvance(String comando)
{
    if(comando.equalsIgnoreCase("VIAJARS"))
    {
        if(orientacion.equalsIgnoreCase("oeste"))
        {
            concatenar(1);
        }
    }
    else if(comando.equalsIgnoreCase("VIAJARW"))
    {
        if(orientacion.equalsIgnoreCase("oeste"))
        {
            concatenar(3);
        }

    }
    else if(comando.equalsIgnoreCase("VIAJARN"))
    {
        if(orientacion.equalsIgnoreCase("oeste"))
        {
            concatenar(2);
            concatenar(3);
        }
    }
```

Fig. 10. Módulo de bajo nivel como servicio web

En este caso se determinó que el plan se debería traducir a una serie de números que varía según la localización actual del robot en cada paso ejecutado y en la que se consideró la siguiente correspondencia entre las acciones y los valores generados: Para la acción girar a la izquierda, se escribiría un 1 en la secuencia; para girar a la derecha, un 2; para avanzar, un 3; para retroceder, un 4; para agarrar un pilote, un 5 y para soltarlo un 6, quedando así:

Tabla 1: Valores de traducción para el plan

Acción	Valor traducción
Giro izquierda	1
Giro derecha	2
Adelante	3
Atrás	4
Agarrar	5
Soltar	6

De este modo, el plan de alto nivel es llevado a una serie de números que son enviados posteriormente al robot y ejecutados por este según la acción asociada.

- **Programa de control y comunicación**

Este programa es el encargado de recibir por parámetros cada uno de los números generados en el módulo de traducción y llamar a las funciones correspondientes según el valor recibido.

```
for( int i = 0; i < args.length; i++)
{
    String s = args[i];
    if(s.compareTo("1") == 0)
    {
        izquierda();
    }
    else if(s.compareTo("2") == 0)
    {
        derecha();
    }
    else if(s.compareTo("3") == 0)
    {
        adelante();
    }
    else if(s.compareTo("4") == 0)
    {
        atras();
    }
    else if(s.compareTo("5") == 0)
    {
        agarrar();
    }
    else if(s.compareTo("6") == 0)
    {
        soltar();
    }
}
```

Fig. 11. Programa de control y comunicación con el robot

Cabe resaltar que este programa no forma parte de la arquitectura PELEA, éste es un programa que es llamado desde el módulo ejecutor de manera externa (similar a como se hizo en el módulo de planificación).

4. DESCRIPCIÓN SEMÁNTICA DE LOS SERVICIOS WEB IMPLEMENTADOS

Con el fin de que las máquinas puedan entender inequívocamente lo que hace cada servicio, se optó por realizar la respectiva descripción semántica para cada uno de los servicios implementados en la sección anterior con el uso de ontologías (Gruber, 2007). La descripción semántica de todos los servicios ha sido creada de manera similar. A continuación se muestra la descripción semántica del módulo ejecutor, lo cual permite su invocación posteriormente:

```
<owl:Ontology rdf:about="">
  <owl:imports rdf:resource="&local;Individuals/Individuals.owl"/>
  <owl:imports rdf:resource="&local;Domain/Domain.owl"/>
  <owl:imports rdf:resource="&daml;Grounding.owl"/>
  <owl:imports rdf:resource="&daml;Profile.owl"/>
</owl:Ontology>
<grounding:WsdlOperationRef rdf:ID="WsdlOperationRef_10">
  <grounding:operation
    rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
      &local2;PeleaExecutorService?WSDL#peleaExecutor
  </grounding:operation>
  <grounding:portType
    rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
      &local2;PeleaExecutorService?WSDL#PeleaExecutorPort
  </grounding:portType>
</grounding:WsdlOperationRef>
```

Fig. 12. Descripción semántica del servicio web ejecutor

Como se puede apreciar en la Fig. 12, el servicio web hace un importe de la ontología que contiene la descripción de las entradas y salidas de los módulos de la arquitectura PELEA, al igual que de la ontología que contiene las instancias del DomainH y el Problem, con el fin de facilitar la obtención de estos en el momento de la invocación del servicio web, enviando como parámetros dichas instancias. También se puede apreciar que tanto las entradas y salidas del servicio web, están descritas como tipos de datos complejos pertenecientes a la ontología Domain.owl.

5. INVOCACIÓN DE LOS SERVICIOS A PARTIR DE SU DESCRIPCIÓN SEMÁNTICA.

Al tener una descripción semántica de los servicios, su invocación se puede realizar por medio de la API de OWL-S (Siren y Parsia, 2004). Esta API, permite crear una base de conocimiento soportada por las instancias de las ontologías de los servicios web y del dominio. Esta API, basada en la descripción semántica de cada servicio web, permite obtener una instancia del servicio y del proceso asociado con éste, se definen las entradas (complejas o primitivas), las salidas requeridas por estos servicios y también permite la ejecución del proceso asociado con cada servicio. En la Fig. 13 se muestran apartes del código en Java que, cuando se ejecuta mediante la API OWL-S, hace la invocación del servicio ejecutor del sistema.

```
public class PeleaInvoker
{
    public static void main(String[] args)
        throws IOException, ExecutionException
    {
        ProcessExecutionEngine exec;
        exec = OWLSFactory.createExecutionEngine();

        OWLKnowledgeBase kb = OWLFactory.createKB();
        String base;
        base = "http://localhost/PAE_2_2012/";
        String ejec;
        ejec = base+"Executor/ExecutorOntology.owl";
        URI url = URI.create(ejec);
        Service aService = kb.readService(url);
        Process aProcess = aService.getProcess();
        ValueMap<Input, OWLValue> inputs;
        inputs = new ValueMap<Input, OWLValue>();
        OWLIndividualList<Input> inputs1;
        inputs1 = aProcess.getInputs();
        String indiv = base+"Individuals/Individuals.owl#";
        URI p = URI.create(base+indiv+"Problem_1");
        URI dH = URI.create(base+indiv+"DomainH_1");
        URI dL = URI.create(base+indiv+"DomainL_1");
        inputs.setValue(inputs1.get(1), kb.getIndividual(p));
        inputs.setValue(inputs1.get(2), kb.getIndividual(dH));
        inputs.setValue(inputs1.get(0), kb.getIndividual(dL));
        ValueMap<Output, OWLValue> outputs
        outputs = exec.execute(aProcess, inputs, kb);
        String out;
        out = outputs.getStringValue(aProcess.getOutput());
        System.out.println(out);
    }
}
```

Fig. 13. Invocación del módulo ejecutor del sistema

6. EVALUACIÓN

Con el fin de dar una correcta validación al proyecto, se procede a evaluar el comportamiento del sistema planteado para una cantidad de pilotes que varía en cada repetición. Para dicha evaluación se plantearon seis situaciones cambiando el número de pilotes en cada una y tomando el tiempo y el número de acciones que debió ejecutar el robot para solucionar el problema por medio del sistema aquí planteado, obteniendo lo siguiente:

Tabla 2. Resultados Evaluación

Nro. Pilotes	Nro. de acciones	Tiempo (min:seg.ms)
5	63	9:17.89
4	69	9:15.60
3	73	8:22.88
2	77	9:42.23
1	85	10:01.30
0	71	7:39.71

Originalmente se esperaría que a mayor número de acciones ejecutadas por el robot mayor el tiempo utilizado para solucionar el problema. Sin embargo, hay que tener en la cuenta que hay acciones que toman más tiempo que otras (el tiempo de giro es mayor al tiempo que toma avanzar una casilla). Es por esto último que se aprecia una variación en los tiempos de ejecución, pues se esperaría que a medida que aumenta el número de pilotes aumente dicho tiempo pero nuevamente, dado que hay acciones más costosas en tiempo que otras, no se puede asegurar un menor tiempo con un menor número de pilotes, pues hay casos como lo son el de uno y dos pilotes que aumentan su tiempo significativamente con respecto de los demás dado que requieren un mayor número de giros que los otros.

7. CONCLUSIONES Y TRABAJO FUTURO

El uso de la arquitectura PELEA facilita la modularización del sistema implementado facilitando la sostenibilidad y mantenimiento del mismo, dicha modularidad permite su implementación de diferentes formas como servicios web en este caso.

Con la implementación del prototipo del sistema de manipulación remota de robots por medio de servicios web semánticos, los usuarios, tanto humanos como las máquinas, pueden tener acceso al robot desde cualquier parte siempre y cuando se tenga un punto de acceso a la web. Además, como se contempla una descripción semántica de los servicios que manipulan el robot, es posible realizar tareas de manera automática ya que las máquinas entienden la funcionalidad de los servicios que abren opciones interesantes como el descubrimiento automático de tareas que hace el robot, al igual que componer automáticamente nuevas tareas que combinen los servicios asociados con las tareas que

hace el robot.
Se encontró que existe la API OWL-S, que es una herramienta adecuada para invocar fácilmente servicios web marcados semánticamente. Esta API presenta un adecuado acoplamiento con las descripciones de un dominio hechas en OWL y con las descripciones de servicios web con la ontología OWL-S. Su programación se hace en Java y se puede editar, invocar y crear servicios web desde cero.

El uso de servicios web para implementar en ellos los procesos complejos asociados con las tareas que debe realizar un sistema robótico y deja la implementación de pequeñas tareas básicas en el propio hardware del robot, permite que se abarate el costo de los robots porque no se requeriría un hardware altamente complejo en el propio robot.

Por último, la metodología propuesta en este trabajo es una muestra de cómo podría ser el proceso de construcción basado en una secuencia de capas que involucre la manipulación remota de un robot cualquiera.

Como trabajo futuro se propone hacer, de manera experimental, pruebas que permitan evaluar la complejidad y la eficiencia de esta clase de sistemas robóticos basados en servicios web semánticos orientados al control remoto de los robots.

Otro trabajo futuro que se tiene proyectado es la implementación de un sistema capaz de componer tareas robóticas complejas a partir de tareas descritas mediante los servicios web semánticos. Para ello, se piensa utilizar algunas herramientas basadas en técnicas de planificación de inteligencia artificial adaptadas en la composición de servicios web.

Como última línea de trabajo futuro se plantea trabajar en mejorar nuevas técnicas de comunicación remota entre el servidor y el propio robot. Para ello se plantea estudiar nuevas tecnologías de comunicación inalámbrica como WI-FI y hacer uso de redes de telefonía celular que permitan dar un mayor alcance entre el servidor y el robot.

RECONOCIMIENTO

Este trabajo es apoyado por el proyecto “Programa de Fortalecimiento del Grupo de Investigación Sistemas Inteligentes Web – Sintelweb, Convocatoria Nacional 2010-2012, Modalidad 3”, con código 20201009532.

REFERENCIAS

Alcázar, V. Guzmán, C. Prior, D. Borrego, D. Castillo, L. Oneida, E. (2010). “PELEA: Planning, Learning and Execution Architecture”.

Boyce, S., & Pahl, C. (2007). Developing Domain Ontologies for Course Content. Educational Technology & Society, 10 (3), 275-288.

Brusa, G. Caliusco, L. Chiotti, O. (2006). “A Process for Building a Domain Ontology: an Experience in Developing a Government Budgetary Ontology”.

Cavanaugh, E. (2006). “Web services: Benefits, challenges, and a unique, visual development solution”, white paper, Sitio web de Altova. [En línea], Disponible en:
http://www.altova.com/whitepapers/webservice.pdf

Ceravolo, P. Damiani, E. Fugazza, C. (2006). “A Transfusion Ontology for Remote Assistance in Emergency Health Care”, On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops, pp.1044–1049.

Gala, M. Howe, A. Knoblock, C. McDermott, D. Ram, A. Volos, M. Weld, D. Wilkins, D. (2003). “PDDL-The Planning Domain Definition Language”.

Gerevini, A & Serina, I. (2002). "LPG: a Planner based on Local Search for Planning Graphs". Sixth International Conference on Artificial Intelligence Planning and Scheduling (AIPS'02), AAAI Press, Toulouse, France.

Gruber, T. (2007). “What is an Ontology”.

Gunner, H. (2002). "Introduction to Web Services".

Hestand, P. (2011). A service oriented architecture for robotic platforms, University of Massachusetts Lowell.

Hu, G. Toy, W. Wen, Y. (2011). "Cloud Robotics: Architecture, Challenges and Applications”.

IEEE. (2011). “Concurso Latinoamericano de Robótica para Estudiantes”, [en línea], Disponible en:http://ewh.ieee.org/reg/9/robotica/Reglas/reglas_eng2011.htm

Lafuente, A. Larrea, M. (2009). "Sistemas Distribuidos. Introducción", Dpto.ATC UP/EHU, Universidad del País Vasco. [En línea], Disponible en:http://www.sc.ehu.es/acwlaalm/sdi/introduccioi-slides.pdf.

Linglom. (2007). "How to run command-line or execute external application from Java", [en línea], Disponible en:
http://www.linglom.com/2007/06/06/how-to-run-command-line-or-execute-external-application-from-java/

Martin, D. Paolucci, M. McIlraith, S. Burstein, M. McDenitt, D. McGuinness, D. Parsia, B. Payne, T. Sabou, M. Solanki, M. Srinivasan, N. Sycara, K. (2004). "Briging Semantic to Web Services: The OLW-S Approach".

Siren E & Parsia B. (2004). "The OWL-S Java API".

UDI. (2012). "Encuentro Internacional de Robótica Interuniversitaria" [en línea], Disponible en: http://www.udi.edu.co/robotica3.0/Descargas/Reglas%20Robot%20Constructor%20de%20Diques.pdf

Vaculin, R. Sycara, K. (2007). "Semantic Web Services Monitoring: AN OWL-S based Approach".

WillowGarage. (2011). "Personal Robot 2 (PR2)," [en línea], Disponible en: www.willowgarage.com.