ISSN: 1692-7257 - Volumen 2 - Número 14 - Año 2009

XSPDDL: A XML BASED LANGUAGE FOR PLANNING DOMAIN DEFINITION ORIENTED WEB

XSPDDL: UN LENGUAJE BASADO EN XML PARA LA DEFINICION DE DOMINIOS DE PLANIFICACION ORIENTADOS A LA WEB

MSc. Jaime Alberto Guzmán Luna*, Ing. Daniel Alonso Areiza Zabala*
MSc. Ailín Orjuela Duarte**

*** SINTELWEB: Grupo de Investigación "Sistemas Inteligentes Web"**
Escuela de Sistemas, Universidad Nacional de Colombia – Sede Medellín
Calle 59A No 63 – 20, Medellín, Colombia
{jaguzman, daareiza}@unal.edu.co
**** Grupo CICOM "Ciencias Computacionales" Universidad de Pamplona**
Ciudadela Universitaria. Pamplona, Norte de Santander, Colombia.
Tel.: 57-7-5685303, Fax: 57-7-5685303, Ext. 158
E-mail: aorjuela@unipamplona.edu.co

Abstract: This paper defines the syntax and semantics of XSPDDL, an XML-based language used by INDIGO (a Web service composition system) and BAXSET (a service semantic search engine). The ability to handle the basic structure of XML makes it compatible with other Web languages for defining the preconditions and effects associated with semantic Web services. Also, its features allow it to specify planning domains oriented Web.

Resumen: Este documento define la sintaxis y semántica del XSPDDL, un lenguaje basado en XML utilizado por INDIGO (un sistema de composición de servicios Web) y BAXSET (un buscador semántico de servicios). La habilidad de manejar las estructuras básicas del lenguaje XML lo hace compatible con otros lenguajes Web utilizados para la definición de las precondiciones y efectos asociados a los servicios Web semánticos. Así mismo, sus características le permiten especificar dominios de planificación orientados a la Web.

Keywords: PDDL, XML, semantic web services, automatic planning, artificial intelligence.

1. INTRODUCCIÓN

Los servicios Web Semánticos (SWS) (Web services HP, 2007), son servicios Web cuyas descripciones internas y externas están en un lenguaje basado en XML (XML, 2001) y tiene semánticas bien definidas interpretables por las máquinas (McIlraith et al, 2001).

Un tópico importante de los SWS es la composición de servicios. Para soportar la composición de servicios, se han propuesto diversas técnicas provenientes de la planificación Automática (Nau et al., 2004) las cuales asocian los SWS a las acciones de un problema de planificación, realizando para ello una traducción de sus respectivas especificaciones al lenguaje específico de planificación.

En este documento, se presenta el XSPDDL, un Esquema XML (XML, 2001) del PDDL que permite simplificar el parseo, la lectura y la comunicación SOAP usada por los SWS y que dada su habilidad para manejar los espacios de nombres (Namespace, 2006) con URIs (Adressing, 2001) y prefijos lo hace altamente compatible con los lenguajes de especificación de los SWS.

Este documento está organizado de la siguiente manera: en la sección 2, se presenta una visión general de las características que implementa el PDDL; en la sección 3 se describe el XSPDDL donde se detalla su sintaxis y su semántica; en la sección 4, se detallan los principales componentes del lenguaje mediante su aplicación en un dominio específico; en la sección 5, se compara el XSPDDL con los principales trabajos del estado del arte que dirigen sus esfuerzos a generar una versión de PDDL mas cercana a los lenguajes usados para especificar los SWS.

2. UNA VISION GENERAL DEL PDDL

La comunidad de la planificación IA en general utiliza formalismos diferentes para expresar los dominios de planificación. Uno de estos formalismos es el PDDL (*Planning Domain Definition Language*) (Fox y Long, 2003). El PDDL, fue desarrollado como el lenguaje de entrada de los planificadores utilizados para la Competición Internacional de Planificación (IPC) desde 1998. A lo largo de las diferentes IPC, el PDDL ha evolucionado para cubrir las necesidades de representación de los nuevos retos formulados en dicho evento.

El PDDL1.2 (utilizado en el IPC-1998 e IPC-2000) contenía el STRIPS (Fikes y Nilsson, 1971) y la funcionalidad del ADL (Pednault, 1989) más el uso de variables con tipo. El PDDL2.1 (IPC-2002) aumentó la versión original del PDDL con: (i) Las variables numéricas y la capacidad de evaluar y actualizar sus valores; y (ii) Las acciones con duración tanto con efectos discretos y continuos. El PDDL2.2 (IPC-2004) amplió las versiones anteriores con el manejo de: (i) predicados derivados que le permiten al planificador razonar sobre conceptos de alto nivel en el dominio, donde tales conceptos pueden ser definidos de forma recursiva y (ii) literales iniciales temporales, los cuales son literales que serán verdaderos en un tiempo predecible independiente de lo que el agente de planificación hace. El PDDL3.0 (IPC-2006) enriquece la expresividad del lenguaje para definir: (i) las restricciones en las transiciones del estado, (ii) las preferencias de objetivos y trayectoria del estado que el plan solución debe seguir. Por último, el PDDL3.1 (IPC-2008) soporta el STRIPS funcional (Geffner, 2001). STRIPS funcional es una codificación diferente para el dominio de la planificación. En lugar de mapear los literales del problema de planificación a verdadero o falso, el STRIPS funcional mapea los objetos del problema de planificación a sus propiedades.

Esta codificación proporciona un modelo más natural para muchos dominios de planificación y hace más fácil la extracción de algunas heurísticas, tal como la heurística del grafo causal (Helmert y Geffner, 2008) o la base de datos de patrones (Edelkamp, 2003).

Aunque el PDDL3.1 cubre todas las anteriores funcionalidades, la mayoría de los planificadores no las implementan. En general, la mayoría solo soportan las características del STRIPS además de las variables con tipo y el predicado de igualdad.

Para brindar un mejor acercamiento a este lenguaje, a continuación se describe la especificación PDDL de uno de los dominios más conocidos dentro de la planificación clásica, el mundo de los bloques (ver Fig. 1).

Este dominio de planificación consiste en alcanzar a partir de de un estado inicial, conformado por un conjunto ordenado de bloques, un estado objetivo, que consiste en un nuevo conjunto ordenado de bloques. Para tal fin, se describe cada una de las acciones sobre los bloques y los respectivos estados inicial y final.

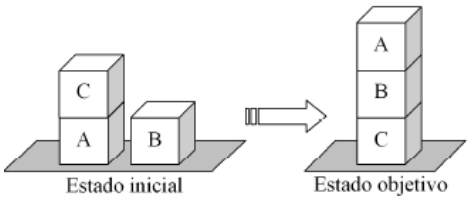


Fig. 1. Problema del Mundo de los Bloques

En general la especificación de un problema de planificación en PDDL consta de la especificación del dominio y de la especificación de su respectivo problema. Para el mundo de los bloques esta especificación podría ser la detallada en la Fig. 2.

Especificación del Dominio	*Especificación del Problema*
(define (domain BLOCKS)	(define (problem Sussman)
(:requirements :strips :typing)	(:domain BLOCKS)
(:types block)	(:objects A B C)
(:predicates (on ?x - block ?y - block) (ontable ?x - block) (clear ?x - block) (handempty) (holding ?x - block))	(:init (on C A) (ontable A) (ontable B) (clear B) (clear C) (handempty))
(:action put-down :parameters (?x - block) :precondition (holding ?x) :effect (and (not (holding ?x)) (clear ?x) (handempty) (ontable ?x)))	(:goal (and (on A B) (on B C))))

Fig. 2. PDDL del Mundo de los Bloques

Basado en la Fig. 2, la especificación del Dominio en PDDL puede constar básicamente de: (i) el nombre del dominio definido mediante la sentencia *domain*; (ii) la especificación de los requerimientos del dominio, definidos mediante la sentencia *requirements*; (iii) Las variables con tipos a utilizar, definidas mediante la sentencia *types*; (iv) los predicados del dominio definidos mediante la sentencia *predicates* y que describen las propiedades de los objetos que conforman el problema (v) las acciones del dominio que se definen mediante la sentencia *action* y detallan la manera en que se cambia el mundo del dominio, esta consta de los parámetros que requiere la acción (*parameters*), las precondiciones que son necesarias para que la acción se lleve a cabo (*precondition*) y los efectos que esta acción produce (*effect*).

Así mismo, basado en la Fig. 2, la especificación del Problema en PDDL puede constar básicamente de: (i) los objetos que conforman el dominio y que están asociados a los diferentes tipos definidos en el dominio (*objects*); (ii) el estado inicial (*init*) el cual detalla el estado actual del mundo que se describe y consta de un conjunto instanciado de predicados; (iii) el estado objetivo al cual queremos llegar (*goal*).

3. EL XSPDDL

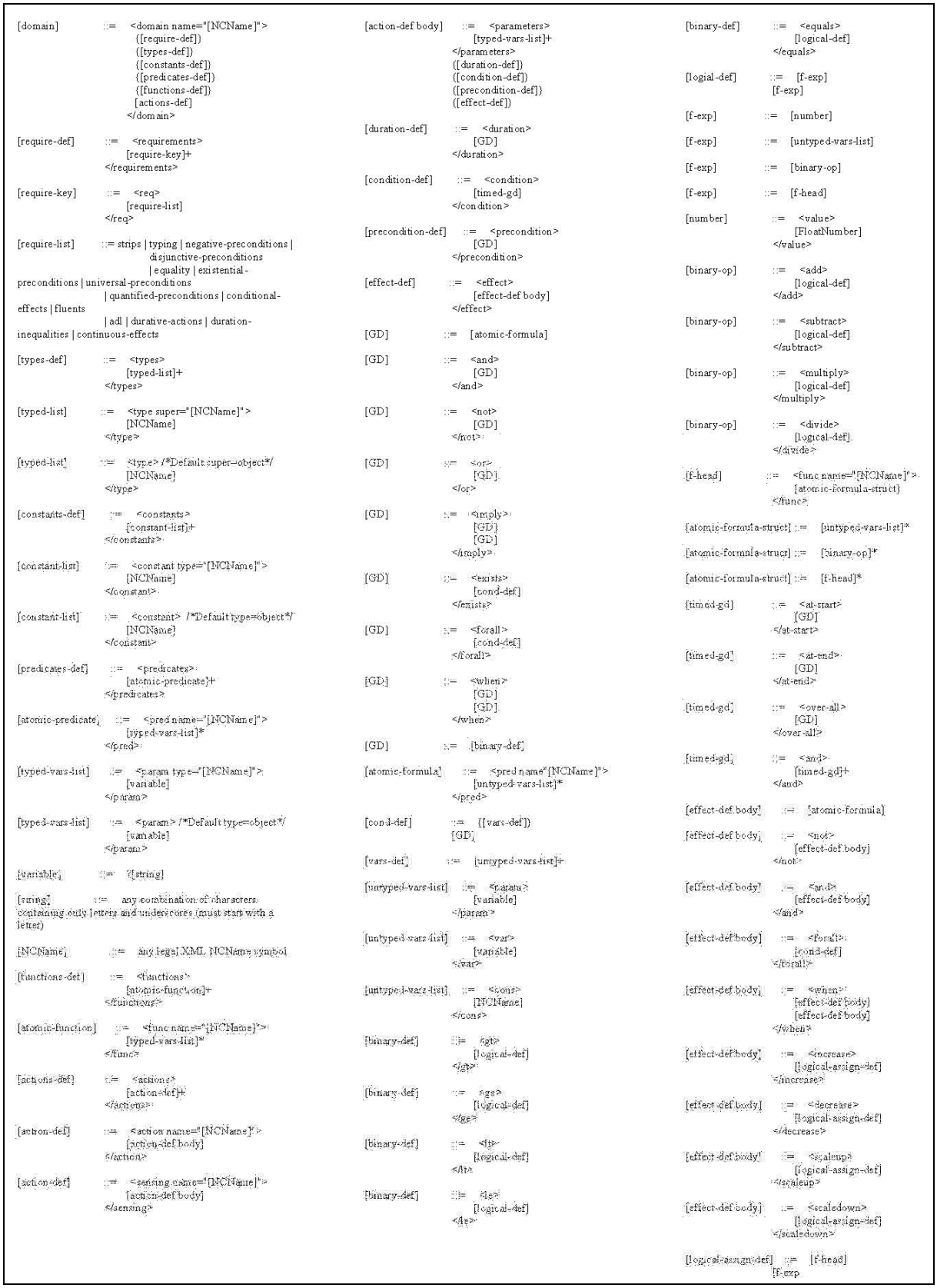
La tecnología XML busca dar solución al problema de expresar información estructurada de la manera más abstracta y reutilizable posible generando documentos que hacen uso de etiquetas. Las etiquetas tienen la forma <*nombre*>, donde *nombre* es el nombre del elemento que se está señalando.

Cada documento XML debe de tener validez, la cual se denomina comúnmente como “que el documento este bien formado”. Que un documento esté "bien formado" solamente se refiere a su estructura sintáctica básica, es decir, que se componga de elementos, atributos y comentarios como XML especifica que se escriban. Esta especificación se encuentra en un documento externo o definición del Esquema XML. Un Esquema XML (XML-Schema) (XML Schema, 2001) permite describir la estructura y las restricciones de los contenidos de los documentos XML de una forma muy precisa, más allá de las normas sintácticas impuestas por el propio lenguaje XML. El principal aporte del Esquema XML es el gran número de tipos de datos que incorpora tales como fechas, números y *strings* entre otros.

En conjunto con XML y el Esquema XML, se desarrolló XPath (Xpath, 1999), el cual es un lenguaje que permite construir expresiones que recorren y procesan un documento XML. XPath permite buscar y seleccionar elementos de un documento teniendo en cuenta la estructura jerárquica del XML que es procesada por un analizador (o parser) construyendo un árbol de nodos.

El XSPDDL, es una versión del PDDL2.1 basada en un Esquema XML, que permite definir tanto la estructura del dominio de un problema de planificación (*domain* en PDDL), como su respectiva especificación del problema (*problem* en PDDL). Con el fin de definir la sintaxis del XSPDDL, a continuación primero se definen la lista de símbolos: (i) paréntesis angulares (<, >) identifica las etiquetas de un elemento XML; (ii) paréntesis Cuadrados ([,]) delimita los nombres de los elementos sintácticos; (iii) las llaves ({, }) representa un elemento adicional; (iv) el asterisco (*) significa "cero o más de" y (v) un más (+) significa "uno o más de".

Así la sintaxis del XSPDDL que permite representar un dominio se detalla en Fig. 3 y la sintaxis del XSPDDL que permite representar un problema se detalla en Fig. 4. En cuanto a su semántica, el XSPDDL se basa en la semántica definida en la versión del PDDL2.1 (Fox y long, 2003) para cada una de sus estructuras, extendiéndola al considerar acciones de sensado (*sensing-action*) para adquirir información del entorno.



```
[domain]          ::=  <domain name="[NCName]">
                         ([require-def])
                         ([types-def])
                         ([constants-def])
                         ([predicates-def])
                         ([functions-def])
                         [actions-def]
                       </domain>

[require-def]     ::=  <requirements>
                         [require-key]+
                       </requirements>

[require-key]     ::=  <req>
                         [require-list]
                       </req>

[require-list]    ::= strips | typing | negative-preconditions |
                      disjunctive-preconditions
                      | equality | existential-
preconditions | universal-preconditions
                      | quantified-preconditions | conditional-
effects | fluents
                      | adl | durative-actions | duration-
inequalities | continuous-effects

[types-def]       ::=  <types>
                         [typed-list]+
                       </types>

[typed-list]      ::=  <type super="[NCName]">
                         [NCName]
                       </type>

[typed-list]      ::=  <type> /*Default:super=object*/
                         [NCName]
                       </type>

[constants-def]   ::=  <constants>
                         [constant-list]+
                       </constants>

[constant-list]   ::=  <constant type="[NCName]">
                         [NCName]
                       </constant>

[constant-list]   ::=  <constant> /*Default:type=object*/
                         [NCName]
                       </constant>

[predicates-def]  ::=  <predicates>
                         [atomic-predicate]+
                       </predicates>

[atomic-predicate]  ::=  <pred name="[NCName]">
                           [typed-vars-list]*
                         </pred>

[typed-vars-list]  ::=  <param type="[NCName]">
                          [variable]
                        </param>

[typed-vars-list]  ::=  <param> /*Default:type=object*/
                          [variable]
                        </param>

[variable]        ::=  ?[string]

[string]          ::=  any combination of characters
containing only letters and underscores (must start with a
letter)

[NCName]          ::=  any legal XML NCName symbol

[functions-def]   ::=  <functions>
                         [atomic-function]+
                       </functions>

[atomic-function]  ::=  <func name="[NCName]">
                          [typed-vars-list]*
                        </func>

[actions-def]     ::=  <actions>
                         [action-def]+
                       </actions>

[action-def]      ::=  <action name="[NCName]">
                         [action-def body]
                       </action>

[action-def]      ::=  <sensing name="[NCName]">
                         [action-def body]
                       </sensing>

[action-def body]  ::=  <parameters>
                          [typed-vars-list]+
                        </parameters>
                        ([duration-def])
                        ([condition-def])
                        ([precondition-def])
                        ([effect-def])

[duration-def]    ::=  <duration>
                         [GD]
                       </duration>

[condition-def]   ::=  <condition>
                         [timed-gd]
                       </condition>

[precondition-def]  ::=  <precondition>
                           [GD]
                         </precondition>

[effect-def]      ::=  <effect>
                         [effect-def body]
                       </effect>

[GD]              ::=  [atomic-formula]

[GD]              ::=  <and>
                         [GD]
                       </and>

[GD]              ::=  <not>
                         [GD]
                       </not>

[GD]              ::=  <or>
                         [GD]
                       </or>

[GD]              ::=  <imply>
                         [GD]
                         [GD]
                       </imply>

[GD]              ::=  <exists>
                         [cond-def]
                       </exists>

[GD]              ::=  <forall>
                         [cond-def]
                       </forall>

[GD]              ::=  <when>
                         [GD]
                         [GD]
                       </when>

[GD]              ::=  [binary-def]

[atomic-formula]  ::=  <pred name"[NCName]">
                         [untyped-vars-list]*
                       </pred>

[cond-def]        ::=  ([vars-def])
                       [GD]

[vars-def]        ::=  [untyped-vars-list]+

[untyped-vars-list]  ::=  <parara>
                            [variable]
                          </param>

[untyped-vars-list]  ::=  <var>
                            [variable]
                          </var>

[untyped-vars-list]  ::=  <cons>
                            [NCName]
                          </cons>

[binary-def]      ::=  <gt>
                         [logical-def]
                       </gt>

[binary-def]      ::=  <ge>
                         [logical-def]
                       </ge>

[binary-def]      ::=  <lt>
                         [logical-def]
                       </lt>

[binary-def]      ::=  <le>
                         [logical-def]
                       </le>

[binary-def]      ::=  <equals>
                         [logical-def]
                       </equals>

[logial-def]      ::=  [f-exp]
                       [f-exp]

[f-exp]           ::=  [number]

[f-exp]           ::=  [untyped-vars-list]

[f-exp]           ::=  [binary-op]

[f-exp]           ::=  [f-head]

[number]          ::=  <value>
                         [FloatNumber]
                       </value>

[binary-op]       ::=  <add>
                         [logical-def]
                       </add>

[binary-op]       ::=  <subtract>
                         [logical-def]
                       </subtract>

[binary-op]       ::=  <multiply>
                         [logical-def]
                       </multiply>

[binary-op]       ::=  <divide>
                         [logical-def]
                       </divide>

[f-head]          ::=  <func name="[NCName]">
                         [atomic-formula-struct]
                       </func>

[atomic-formula-struct]  ::=  [untyped-vars-list]*

[atomic-formula-struct]  ::=  [binary-op]*

[atomic-formula-struct]  ::=  [f-head]*

[timed-gd]        ::=  <at-start>
                         [GD]
                       </at-start>

[timed-gd]        ::=  <at-end>
                         [GD]
                       </at-end>

[timed-gd]        ::=  <over-all>
                         [GD]
                       </over-all>

[timed-gd]        ::=  <and>
                         [timed-gd]+
                       </and>

[effect-def body]  ::=  [atomic-formula]

[effect-def body]  ::=  <not>
                          [effect-def body]
                        </not>

[effect-def body]  ::=  <and>
                          [effect-def body]
                        </and>

[effect-def body]  ::=  <forall>
                          [cond-def]
                        </forall>

[effect-def body]  ::=  <when>
                          [effect-def body]
                          [effect-def body]
                        </when>

[effect-def body]  ::=  <increase>
                          [logical-assign-def]
                        </increase>

[effect-def body]  ::=  <decrease>
                          [logical-assign-def]
                        </decrease>

[effect-def body]  ::=  <scaleup>
                          [logical-assign-def]
                        </scaleup>

[effect-def body]  ::=  <scaledown>
                          [logical-assign-def]
                        </scaledown>

[logical-assign-def]  ::=  [f-head]
                           [f-exp
```

Fig. 3. Sintaxis BNF del XSPDDL para la definición del dominio

```
[domain]        ::=  <problem name="[NCName]">
                       [domain-def]
                       {[objects-dec]}
                       [init-def]
                       [goal-def]
                       {[metric-def]}
                     </problem>

[domain-def]    ::=  <domain>
                       [NCName]
                     </domain>

[objects-dec]   ::=  <objects>
                       [obj-list]+
                     </objects>

[obj-list]      ::=  <obj type="[NCName]">
                       [NCName]
                     </obj>

[obj-list]      ::=  <obj> /*Default type=object*/
                       [NCName]
                     </obj>

[init-def]      ::=  <init>
                       [init-list]+
                     </init>

[init-list]     ::=  [atomic-formula]

[init-list]     ::=  <not>
                       [atomic-formula]
                     </not>

[init-list]     ::=  <assign>
                       [f-head]
                       [number]
                     </assign>

[goal-def]      ::=  <goal>
                       [GD]
                     </goal>

[metric-def]    ::=  <metric>
                       [metric-opt]
                     </metric>

[metric-opt]    ::=  <minimize>
                       [ground-f-exp]
                     </minimize>

[metric-opt]    ::=  <maximize>
                       [ground-f-exp]
                     </maximize>

[ground-f-exp]  ::=  [f-exp]

[ground-f-exp]  ::=  <total-time />
```

Fig. 4. Sintaxis BNF del XSPDDL para la definición del problema

4. UN EJEMPLO DEL XSPDDL

Con el fin de dar una mejor claridad de la aplicación del XSPDDL a continuación se describen las principales estructuras que conforman el dominio y el problema del mundo de los bloques usando este lenguaje. La primera estructura que hace parte de la definición del dominio, es la estructura *domain*. Esta estructura contiene el atributo *name* para especificar el nombre del dominio. Como se detalla en Fig 3, esta es la estructura que contiene los demás elementos que componen la definición del dominio:

PDDL	(define (domain BLOCKS)
XSPDDL	<domain name="BLOCKS">

Veamos ahora la estructura de los *requirements*. Esta estructura define los requerimientos del dominio, una lista de elementos *req* que contienen un *string* con el tipo de requerimiento a utilizar por el planificador:

PDDL	(:requirements :strips :typing)
XSPDDL	<requirements> <req>strips</req> <req>typing</req> </requirements>

La estructura siguiente, *type*. Esta estructura define los tipos del dominio utilizando en su interior una lista de elementos *type*. Estos últimos, contienen a su vez un *string* con el tipo a utilizar por el planificador*:*

PDDL	(:types block)
XSPDDL	<types> <type>block</type> </types>

La siguiente estructura, que vale la pena detallar es la estructura *predicates*. Esta estructura está compuesta por una lista de *pred*, que representan los predicados asociados al dominio. Estos a su vez contienen una lista de *param* que permiten especificar el tipo del parámetro por medio de su atributo *type* y el nombre del mismo parámetro mediante el uso de un *string* contenido en su interior:

PDDL	(:predicates (on ?x - block ?y - block)
XSPDDL	<predicates> <pred name="on"> <param type="block">x</param> <param type="block">y</param> </pred>

Por último, se tiene la estructura *actions*. Dada su complejidad, con el fin de dar una mejor claridad de la misma, se dividirá en sus sub-estructuras. La primera sub-estructura es la estructura *parameters*, la cual solo es una lista de etiquetas de tipo *param* que representan los parámetros que serán utilizados por la acción:

PDDL	(:action pick-up: parameters (?x - block)
XSPDDL	<actions> <action name="pick-up"> <parameters> <param type="block">x</param> </parameters>

La siguiente sub-estructura es la *preconditions*:

PDDL	:precondition (and (clear ?x) (ontable ?x) (handempty))
XSPDDL	<preconditions> <and> <pred name="clear"> <param>x</param> </pred> <pred name="ontable"> <param>x</param> </pred> <pred name="handempty" /> </and> </preconditions>

Para el caso particular del mundo de los bloques, la anterior especificación contiene una estructura *and*, la cual a su vez contiene una lista de etiquetas de tipo *pred* que representan los predicados asociados a las precondiciones que se deben satisfacer para que la acción se pueda ejecutar.

Por último, la sub-estructura *effects*. En el caso del mundo de los bloques esta se compone de una estructura *and* la cual contiene una estructura *pred* y una estructura *not*, etiqueta que especifica un predicado negado:

PDDL
```
:effect
(and
  (not (ontable ?x))
  (not (clear ?x))
  (not (handempty))
  (holding ?x)))
```

XSPDDL
```
<effects>
<and>
  <not>
    <pred name="ontable">
      <param>x</param>
    </pred>
  </not>
  :
  <pred name="holding">
    <param>x</param>
  </pred>
</and>
</effects>
```

En cuanto a las estructuras utilizadas para la definición del problema este está inicialmente conformado por la estructura *problem*:

PDDL
```
(define (problem Sussman)
    (:domain BLOCKS)
```

XSPDDL
```
<problem name="Sussman" >
    <domain>BLOCKS</domain>
    :
<problem>
```

La anterior estructura esta conformada por un atributo *name*, que define el nombre del problema y contiene en su interior una etiqueta *domain* en cuyo interior se detalla el nombre del dominio asociado al problema. Este último elemento es de tipo *string*:

Para definir los objetos presentes en el problema, se utiliza la estructura *objects.* Esta contiene una lista de etiquetas tipo *obj*, que permiten describir los objetos que conformaran el problema de planificación. Cada *obj* contiene un elemento tipo *string* que permite describir el elemento (por ejemplo A, B C):

PDDL
```
(:objects A B C)
```

XSPDDL
```
<objects>
  <obj>A</obj>
  <obj>B</obj>
  <obj>C</obj>
</objects>
```

Para definir el estado inicial del problema con XSPDDL se utiliza su estructura *init*. Esta estructura esta conformada por una lista de etiquetas de tipo *pred*, que representan los predicados instanciados del problema. Cada *pred* contiene el atributo *name* que permite asignarle el nombre del predicado y en su interior, cada *pred* contiene una lista de etiquetas de tipo *obj*:

PDDL
```
(:init (on C A)
```

XML
```
<init>
  <pred name="on">
    <obj>C</obj>
    <obj>A</obj>
  </pred>
  :
</init>
```

Por último, se utiliza la estructura *goal*, la cual permite especificar el estado objetivo del problema. Para este dominio, el *goal* contiene una etiqueta de tipo *and*, la cual contiene una lista de elementos de tipo *pred* que representan los predicados deseados en el estado objetivo:

PDDL
```
(:goal (and (on A B) (on B C))))
```

XML
```
<goal>
  <and>
    <pred name="on">
      <obj>A</obj>
      <obj>B</obj>
    </pred>
    <pred name="on">
      <obj>B</obj>
      <obj>C</obj>
    </pred>
  </and>
</goal>
```

5. TRABAJOS RELACIONADOS

El XSPDDL ha sido utilizado tanto en la definición de servicios Web Semánticos OWL-S (OWL-s, 2007) como en la definición del problema de composición del sistema INDIGO (Guzmán y Ovalle, 2008), un modelo de planificación de composiciones de servicios Web. Así mismo, este lenguaje es utilizado en el "Sistema de Recuperación Semántico de Servicios Web Basado en un Modelo de Navegación Taxonómica" (BAXSET, 2009), para especificar los SWS de tipo OWL-S.

Al interior de INDIGO y de BAXET, el XSPDDL es utilizado para la especificación de las precondiciones y efectos de los SWS. Para tal fin, se ha extendido la ontología *expression* que hace parte del OWL-S para incluir en ella el lenguaje

XSPDDL. Así mismo, al interior de los dos anteriores mecanismos, se ha desarrollado un analizador sintáctico basado en el modelo DOM (DOM, 2005) que verifica la correctitud de las especificaciones XSPDDL al igual que se ha desarrollado específicamente en INDIGO un traductor de OWL-S a XSPDDL que permite especificar un problema de composición de servicios Web como un problema de planificación en XSPDDL, lenguaje utilizado por el planificador que implementa este modelo.

Actualmente existe un alto interés en el desarrollo de notaciones para describir servicios Web. Un primer ejemplo de esto es el Web-PDDL, el cual es un lenguaje de ontologías fuertemente tipado basado en una lógica de primer orden extendido desde el PDDL. Este permite expresar espacios de nombres XML que lo hacen compatible con lenguajes de ontologías tales como el RDF y el OWL. La presencia de tipos, le permite realizar la comprobación de tipos durante el razonamiento. En contraste con el XSPDDL, el Web-PDDL es un lenguaje de ontología que requiere de un razonador lógico para su uso, mientras que el XSPDDL esta basado en el XML y el Esquema XML, lo que lo hace independiente del uso de un razonador lógico y simplemente con un simple interprete de XML se puede utilizar fácilmente. Esta cualidad del XSPDDL lo hace más rápido en cuanto al parseo de una especificación y su uso en diferentes áreas de aplicación.

Otra propuesta existente en la literatura es el XPDDL (Gough, 2004). Esta es una versión XML del PDDL2.1. Si bien este trabajo va en la misma dirección del XSPDDL, estos dos lenguajes se diferencian en la conformación de sus estructuras sintácticas que lo conforman. En el caso del XPDDL, utiliza las estructuras básicas del Esquema XML para la definición de los elementos que componen el lenguaje PDDL. En el caso del XSPDDL, adicionalmente a las estructuras básicas del Esquema XML, este utiliza el XPath para definir tareas internas de verificación automáticas entre los elementos que conforman el lenguaje. Esto no es implementado por el XPDDL. Adicionalmente, el XPDDL se basa en la simplificación del mundo cerrado (Golden et al, 1994), es decir, considera que los hechos del mundo que no se encuentran especificados en el modelo son falsos. En contraste con esto el XSPDDL, se basa en el mundo abierto, donde un hecho que no sea considerado en la base de conocimiento del sistema no necesariamente quiere decir que es falso. Para ello el XSPDDL extiende la especificación del PDDL2.1 y considera acciones de sensado que le permiten adquirir información del mundo real, sin que esto implique un cambio de este mundo.

6. CONCLUSIONES Y TRABAJO FUTURO

En este documento se ha propuesto un lenguaje para la definición de dominios de planificación orientado a la Web. Este lenguaje permite implementar la semántica del PDDL, lenguaje estándar de la planificación automática. Este lenguaje a diferencia de otras propuestas se basa en la tecnología XML y el Esquema XML. Adicionalmente permite modelar el mundo bajo el enfoque del mundo abierto, para lo cual extiende el PDDL2.1 utilizando acciones de sensado que permiten adquirir información del mundo en vez de cambiarlo.

El lenguaje XSPDDL ha sido utilizado en aplicaciones del mundo real como el sistema de composición de servicios INDIGO y el buscador semántico de servicios Web BAXSET. En estos ha sido utilizado para especificar las precondiciones y efectos de los SWS al igual que en la especificación de un problema de planificación de composiciones de servicios Web.

Como trabajo futuro, se plantea extender tanto la sintaxis como la semántica de este lenguaje para que permita representar algunas características propias del lenguaje de ontología OWL, como son el manejo de la cardinalidad en sus propiedades, así como sus propiedades *inverseFunctional*, *Symetric* y *Transitive* lo cual permitirá hacerlo mas expresivo.

7. RECONOCIMIENTOS

El presente trabajo está apoyado parcialmente en el proyecto “Un Sistema de Recuperación Semántico de Servicios Web Basado en un Modelo de Navegación Taxonómica” apoyado por la DIME, de la Universidad Nacional de Colombia sede Medellín y la Tesis de Doctorado “Modelo de Planificación y Ejecución Concurrente para la Composición de Servicios Web Semánticos en Entornos Parcialmente Observables”, auspiciada por Colciencias, Universidad Nacional de Colombia, Sede Medellín y el Banco Mundial, enmarcado en el programa de apoyo a la comunidad científica nacional en programas de Doctorado 2004.

REFERENCIAS

Adressing: http://www.w3.org/Addressing/ (2001). Última visita agosto 2009.

BAXSET (2009). *Un Sistema de Recuperación Semántico de Servicios Web Basado en un Modelo de Navegación Taxonómica. Proyecto de Investigación, apoyado y financiado por el Departamento de investigación DIME de la Universidad Nacional de Colombia, sede Medellín, Convocatoria Conmemorativa Darwin, 2009.*

DOM (2005). http://www.w3.org/DOM. Última visita agosto de 2009.

Morales E. (2006). Programación Lógica Inductiva (ILP), http://ccc.inaoep.mx/~emorales/ ultima visita septiembre de 2007.

Edelkamp S. (2003). Taming numbers and durations in the model checking integrated planning system. Journal of Artifiial Intelligence Research (JAIR), 20:195- 238,

Fikes R.E. y Nilsson N.J . (1971) STRIPS: A new approach to the application of theorem proving to problem solving. Artiffcial Intelligence, 2:189-208, 1971.

Fox M. y long D.(2003) PDDL2.1: An extension to PDDL for expressing temporal planning domains. Journal of Artificial Intelligence Research (JAIR), 20:61{124, 2003.

Geffner H. (2001) Functional STRIPS: a more flexible language for planning and problem solving. Logic-Based Artificial Intelligence, Jack Minker (Ed.), Kluwer, 2001.

Gough, J., 2005, "XPDDL: A XML version of PDDL". Available in http://www.cis.strath.ac.uk/ jg/XPDDL/, (accessed in April 2007).

Golden K., Etzioni O., y Weld D.. Omnipotence without omniscience: Eficient sensor management for planning. National Conference on Artificial Intelligence (AAAI), pages 1048-1054, 1994.

Guzmán J. y Ovalle A. (2008). INDIGO: Una Propuesta de Planificación en Inteligencia Artificial para la Composición Automática de Servicios Web Semánticos. VII Jornadas de Ingeniería de Software e Ingeniería del Conocimiento 2008, Guayaquil Ecuador.

Helmert, M., y Geffner, H. (2008). Unifying the causal graph and additive heuristics. In Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling (ICAPS 2008), 140–147.

Hoffmann J., (2003). The Metric-FF Planning System: Translating Ignoring Delete Lists to Numeric State Variables, *Journal of Artificial Intelligence Research*, 20.

IPC (2004). International Planning Competition 2004. http://ipc.icaps-conference.org/. Última visita agosto de 2009.

IPC (2006). http://ipc.icaps-conference.org/. Última visita agosto de 2009.

IPC (2008). International Planning Competition 2004. http://ipc.icaps-conference.org/. Última visita agosto de 2009.

Jiménez S, Fernando Fernandez and Daniel Borrajo, (2006). Inducing non-deterministic actions behavior to plan robustly in probabilistic domains", In: Working notes of the ICAPS'06 Workshop on Planning under Uncertainty and Execution Control for Autonomous Systems.

Jiménez S., (2007) Planning and Learning under Uncertainty, *disertacion para Ph.D., Escuela Politécnica Superior Ingeniería Informática*, Universidad Carlos III de Madrid.

McIlraith S., Son T. C., and Zeng H. (2001). Semantic Web Services, *IEEE Intelligent Systems*,16(2). 46 – 53.

Namespace: http://www.w3.org/TR/REC-xml-names/, año 2006. Última visita Agosto de 2009.

Nau D., Ghallab M. and Traverso P. (2004). Automated Planning: Theory and Practice, *Morgan Kaufman*.

OWL-S. Semantic Markup for Web Services. *http://www.w3.org/Submission/OWLS/, última visita septiembre de 2007.*

Pednault E.P.D. (1989). ADL: exploring the middle ground between STRIPS and the situation calculus. Proceedings of the first international conference on Principles of knowledge representation and reasoning, pages 324-332, 1989

Web services HP, Web Services Concepts: a Technical Overview, HP Document, *http://www.bluestone.com/downloads/pdf/web services tech_overview.pdf, ultima visita septiembre de 2007.*

XMLSchema: http://www.w3.org/XML/Schema (2001). Última visita Agosto 2009.

Xpath: http://www.w3.org/TR/xpath (1999). Última visita, Agosto de 2009.