Mejoramiento de la consistencia entre la sintaxis textual y gráfica del lenguaje de *Semat*

Carlos Mario Zapata Jaramillo, Rafael Esteban Arango Sanchez, Leidy Diana Jiménez Pinzón

***Resumen*—Semat (Software Engineering Method and Theory) es una iniciativa que permite representar prácticas comunes de metodologías ya existentes mediante los elementos de su núcleo, los cuales se describen en términos de un lenguaje. Este lenguaje tiene una sintaxis gráfica y una textual. La sintaxis textual se describe mediante el metalenguaje EBNF (Extended Backus-Naur Form) que se utiliza como notación de gramáticas de libre contexto para describir un lenguaje formal. Sin embargo, la sintaxis textual de los elementos del núcleo en algunos casos presenta inconsistencia con la sintaxis gráfica. Por ello, en este artículo se propone la modificación del lenguaje textual mediante un análisis gramatical al lenguaje de Semat con el fin de lograr una relación consistente entre la sintaxis textual y gráfica de los elementos del núcleo de Semat.**

***Palabras clave*—Análisis gramatical, EBNF, Semat, sintaxis textual.**

Improving the consistency between textual and graphical syntax of the language of *Semat*

***Abstract*— Semat (Software Engineering Method and Theory) is an initiative that allows representing common practices of existing methodologies by its core elements, which are described in terms of a language. This language has a graphical and a textual syntax. The textual syntax is described using meta-language EBNF (Extended Backus-Naur Form), which is used as context-free grammar notation to describe a formal language. However, the textual syntax of core elements in some cases is inconsistent with the graphical syntax. Therefore, in this paper we propose a modification of textual language by parsing the language of Semat in order to achieve a consistent relationship between textual and graphical syntax of the core elements of Semat.**

***Keywords*—Parsing, EBNF, Semat, textual syntax.**

I. Introducción

Semat es una iniciativa que apoya un proceso para redefinir la ingeniería de software con base en una teoría sólida, principios probados y mejores prácticas. A diferencia de otros intentos para crear una teoría general de la ingeniería de software, en Semat se generaliza la ingeniería de software identificando acciones y elementos universales, que se describen mediante un lenguaje sencillo y universal que permite la descripción de las prácticas comunes de metodologías existentes y así lograr que se puedan evaluar, comparar y medir [1]. Su núcleo incluye un grupo de elementos esenciales que son universales para todo esfuerzo de desarrollo de software y extensibles para usos específicos, lo que permite asumir que Semat no se resiste ante nuevas ideas, ya que cualquier metodología se puede representar mediante sus elementos en el núcleo [2].

El lenguaje de Semat posee una sintaxis abstracta, la cual se compone de una sintaxis textual y una sintaxis gráfica. La sintaxis gráfica comprende la representación, una forma visual, de los elementos del núcleo de Semat, mientras que la sintaxis textual, se encuentra descrita en el metalenguaje EBNF (*Extended Backus-Naur Form*), presenta una descripción formal de cada uno de los elementos del núcleo [3].

La notación Backus-Naur (BNF por sus siglas en inglés) se creó inicialmente para describir la sintaxis del lenguaje de programación ALGOL 60 y se utiliza desde entonces como notación para las gramáticas libres de contexto, las cuales permiten describir la estructura sintáctica de muchos (aunque no todos) lenguajes [4].

Tal como se puede ver en la fig. 1, una gramática consta de un conjunto de no-terminales, terminales y una serie de reglas de producción. Un no-terminal se define en una regla de producción, mientras que un terminal es un símbolo del lenguaje que se está definiendo. En una regla de producción, el no-terminal (que aparece en la parte izquierda) se define en términos de una secuencia de símbolos no-terminales y terminales (que se encuentran en la parte derecha) [5].

```
<símbolo> ::= <expresión con símbolos>
```

Figura 1. Expresión BNF

EBNF es un conjunto de expansiones de BNF, por lo cual presenta pequeñas diferencias sintácticas y algunas operaciones adicionales. En ella se incorporan algunos conceptos de la notación sintáctica de Wirth con el propósito de definir la gramática de los lenguajes de programación (lenguajes formales) [6].

En este artículo se propone una revisión de la sintaxis textual de los elementos del núcleo de Semat, con el fin de encontrar las inconsistencias existentes entre las representaciones gráficas de los diferentes elementos y las especificaciones descritas en la sintaxis textual. Existen elementos gráficos a los cuales les faltan conexiones que se definen en el lenguaje textual y existen expresiones que representan de forma incompleta lo que el lenguaje gráfico muestra. Adicionalmente, estos elementos se

Manuscrito recibido el 19 de abril de 2014; aceptado para la publicación el 17 de junio del 2014.

Todos los autores están con la Universidad Nacional de Colombia, sede Medellín, Colombia (correos: {cmzapata, raearangosa, ldjimenezp}@unal.edu.co).

TABLA I.
OPERADORES EBNF

Operador	Descripción
‘ ’	Delimitador de carácter. Por ejemplo: **‘b’**
“ ”	Delimitador de cadenas de caracteres. Por ejemplo: **“>=”**
|	Alternativa. Por ejemplo: **‘b’** | **‘a’**
( )	Delimitadores de agrupamiento. Por ejemplo: **(‘a’ | ‘e’ | ‘i’ | ‘o’ | ‘u’)**
. .	Rango. Por ejemplo: **‘1’. .‘7’**
?	Opcional (cero o una vez). Por ejemplo: **(‘0’. .‘9’)?**
*	Repetición cero o más veces. Por ejemplo: **(‘0’. .‘9’)***
+	Repetición una o más veces. Por ejemplo: **(‘0’. .‘9’)+**
~	Negación. Por ejemplo: **~(‘b’ | ‘a’)**

especifican con invariantes y algunas operaciones adicionales definidas en OCL (*Object Constraint Language*). Sin embargo, las expresiones en OCL de Semat aún presentan problemas de consistencia que se deberían corregir para conseguir un uso adecuado de los elementos del lenguaje y poder representar las prácticas y los métodos. Por las razones anteriores, se toman algunos ejemplos específicos del *Essence* [3] y se realiza la especificación correcta de acuerdo con la sintaxis gráfica propuesta.

Este artículo se organiza de la siguiente manera: en la sección II se presenta el marco teórico que incluye una descripción de Semat, los elementos del núcleo y su lenguaje propio y la descripción de la sintaxis del metalenguaje EBNF; en la sección III se presentan los antecedentes sobre las especificaciones con la sintaxis textual que tiene Semat en algunas de sus representaciones gráficas; en la sección IV se propone una modificación a la especificación de dichos ejemplos tomados del *Essence* y, por último, en la sección V se concluye y se propone el trabajo futuro.

II. MARCO TEÓRICO

A. EBNF *(Extended Backus-Naur Form)*

Una especificación EBNF es un sistema de reglas donde existe sólo una acción primitiva. Es una ecuación sintáctica que permite definir una categoría sintáctica S, mediante una expresión E [6], como se aprecia en la fig. 2. La secuencia “::=” es el metasímbolo de la producción. Una producción se puede considerar como la definición de S en términos de E. <S> es no-terminal y la <E> consiste en una lista de términos sintácticos alternativos que incluyen caracteres y operadores.

```
<S> ::= <E>
```

Figura 2. Especificación EBNF

En la notación EBNF se encuentran los operadores que se definen en la Tabla I [6].

Los términos en una ecuación sintáctica se separan con operadores. Tal como se puede ver en la fig. 3, la expresión E sólo se puede reemplazar con uno y solo un término T, pues el metasímbolo | es una “o” excluyente [6].

$$\langle E \rangle ::= \langle T_1 \rangle \mid \langle T_2 \rangle \mid \ldots \mid \langle T_n \rangle, \quad n > 0$$

Figura 3. Términos sintácticos alternativos

Tal como se puede ver en la fig. 4, cada término T se puede reemplazar con la concatenación de factores [6].

$$\langle T \rangle ::= \langle F_1 \rangle \mid \langle F_2 \rangle \mid \ldots \mid \langle F_n \rangle, \quad n > 0$$

Figura 4. Factores sintácticos

Según se aprecia en la fig. 5, a un factor también se le pueden asignar operadores como: Opción (a), Repetición de cero o más veces (b) y Repetición por lo menos una vez (c) [6].

```
<T> ::= [E]    <T> ::= {E}    <F> ::= [E*]
```

(a) (b) (c)

Figura 5. Opción y Repetición

Una forma de verificación es la construcción del árbol de derivación, el cual permite representar gráficamente la especificación EBNF, planteada desde el elemento inicial (raíz) hasta los elementos terminales (hojas) [6]. Un árbol de derivación que ejemplifica la especificación EBNF se muestra en la fig. 6. La forma gráfica se muestra en la fig. 7.

```
<entero con signo> ::= <signo><entero>
<signo> ::= '+' | '-'
<entero> ::= <dígito><entero> | <dígito>
<dígito> ::= '0' | '1'
```

Figura 6. Ejemplo de especificación EBNF (la tercera regla es recursiva)

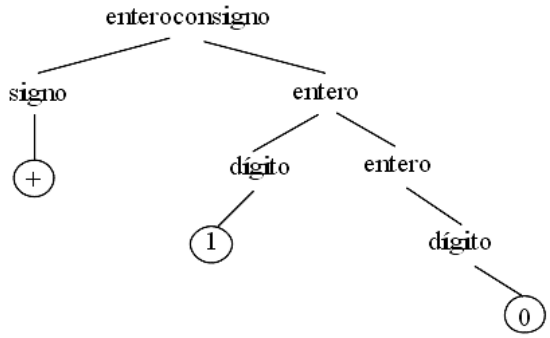

Figura 7. Ejemplo de árbol de derivación

B. Semat (Software Engineering Method and Theory)

La especificación del lenguaje de Semat se constituye utilizando una combinación de tres técnicas diferentes: un metamodelo, un lenguaje formal y lenguaje natural. El metamodelo expresa la sintaxis abstracta y algunas limitaciones

estructurales de las relaciones entre los elementos del núcleo. A raíz de la existencia de una invariante por cada elemento y las limitaciones, se proporcionan reglas de formación del idioma (semántica estática). Los invariantes y algunas operaciones adicionales se establecen utilizando el lenguaje OCL como lenguaje formal. La descripción de los elementos del núcleo y la semántica dinámica se describen mediante lenguaje natural acompañado de definiciones formales utilizando VDM (*Vienna Development Method*) [3].

La sintaxis textual del lenguaje del núcleo de Semat se especifica en EBNF. Además de los operadores descritos anteriormente de esta notación, se identifican dos elementos más [3]:

- ID: palabra específica que representa un identificador para un elemento definido.
- Ref: denota un símbolo que representa un identificador de un elemento (es decir, no del elemento definido).

La sintaxis gráfica del lenguaje de Semat proporciona una forma visual para cada uno de sus elementos, donde cada uno de estos elementos corresponde a un aspecto específico de un núcleo o método. En *Essence* [3] se dividen los elementos de Semat en diferentes categorías. En la fig. 8 se presentan las representaciones gráficas de los elementos descritos a continuación [3].

Grupos de elementos:

- Núcleo: conjunto de elementos usados para formar una base que describe la ingeniería de software.
- Método: es un conjunto de prácticas que forma la descripción de los esfuerzos que se realizan en una empresa. Los esfuerzos se visualizan mediante instancias como alfas, productos de trabajo, actividades y similares.
- Práctica: es un esfuerzo que se puede repetir y se realiza con un propósito específico.

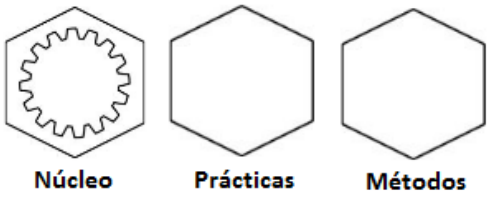

Figura 8. Sintaxis gráfica del grupo de elementos [3]

Elementos del núcleo: en la fig. 9 se presentan las representaciones gráficas de los elementos descritos a continuación [3].

- Alfa: se caracteriza por un conjunto de estados que representan su progreso y salud. Cada estado tiene una lista de chequeo que especifica los criterios necesarios para alcanzar un estado en particular.
- Estado: expresa una situación donde algunas condiciones se proponen.
- Lista de chequeo: lista que se necesita verificar en un estado.
- Espacios de actividad: son las “cosas que siempre hacemos” en ingeniería de software. Agrupan conjuntos de actividades que se realizan mientras se desarrolla un producto de software.
- Competencia: una característica del interesado o equipo que refleja la habilidad de hacer un trabajo. Se puede detallar en diferentes “niveles de competencia”.

Figura 9. Sintaxis gráfica de los elementos del núcleo [3]

Elementos de las prácticas: En la fig. 10 se presentan las representaciones gráficas de los elementos descritos a continuación [3].

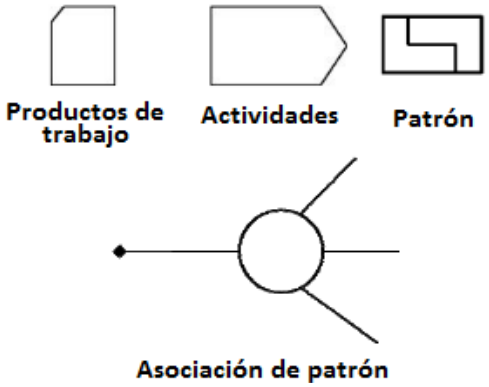

Figura 10. Sintaxis gráfica de los elementos de Semat [3]

- Productos de trabajo: artefactos que se utilizan o se generan en una práctica.
- Actividad: se define como una o más clases de puntos de trabajo y la guía sobre cómo realizarlas.
- Patrón: es una descripción de una estructura en una práctica.
- Asociación de patrón: permite conectar un patrón con otros elementos en una práctica.

III. Antecedentes

En esta Sección se presenta la sintaxis textual de elementos de Semat y la comparación con el lenguaje gráfico. Se propone el estudio de sólo algunos elementos para profundizar acerca de

las ambigüedades encontradas entre los dos tipos de sintaxis que componen su lenguaje.

En el lenguaje textual de Semat se usan tres símbolos globales para la declaración de los elementos:

- ID: se utiliza como identificador del elemento definido y es una palabra única que se utiliza, también, como nombre del elemento.
- Ref, este símbolo denota una referencia a un elemento definido y es la unión del ID con la palabra "Ref" al final.
- STRING, aunque es un tipo de dato; si se usa después del ID se utiliza como una descripción del elemento.

A. *Root Elements*

Dentro de la clasificación de elementos de Semat, existe un grupo denominado "Root elements", que contiene la especificación de los elementos descritos en la Sección anterior y, adicionalmente, propone la organización de otros elementos del lenguaje textual de Semat.

GroupElement:
Kernel | Practice | Library | PracticeAsset | Method;

De acuerdo con la expresión anterior cualquier *Kernel*, *Practice*, *Library*, *PracticeAsset* o *Method* es un *GroupElement*.

A partir del razonamiento de la primera definición, se presentan las siguientes especificaciones que completan el grupo "*Root Elements*":

PatternElement:
Alpha | AlphaAssociation | AlphaContainment |WorkProduct |WorkProductManifest | Activity | ActivitySpace | ActivityAssociation | Competency | Pattern;

Algunos de estos elementos contienen categorías de elementos en su definición, por lo que la cantidad de elementos aumenta, así:

PracticeElement:
PatternElement | ExtensionElement | MergeResolution | UserDefinedType;

Cualquier *PatternElement*, *ExtensionElement*, *MergeResolution* o *UserDefinedType* es un *PracticeElement*, por lo que la definición es equivalente a la siguiente:

PracticeElement:
Alpha | AlphaAssociation | AlphaContainment | WorkProduct |WorkProductManifest | Activity | ActivitySpace | ActivityAssociation | Competency| Pattern | ExtensionElement | MergeResolution | UserDefinedType;

AnyElement:
GroupElement | PracticeElement | State | Level | CheckListItem | CompetencyLevel | PatternAssociation | Tag | Resource;

KernelElement:
Alpha | AlphaAssociation | AlphaContainment | ActivitySpace | Competency | Kernel | ExtensionElement | MergeResolution | UserDefinedType;

StateOrLevel:
State | Level;

AlphaOrWorkProduct:
Alpha | WorkProduct;

AbstractActivity:
Activity | ActivitySpace;

PracticeContent:
PracticeElement | Practice | PracticeAsset;

MethodContent:
Practice | ExtensionElement | MergeResolution;

B. *Element Groups*

En esta sección se profundiza sobre los elementos de esta categoría que presentan inconsistencia con el lenguaje gráfico.

Kernel:
'kernel' ID ':' STRING
('with rules' STRING)?
('owns' '{' KernelElement* '}')?
('uses' '{' KernelElementRef (',' KernelElementRef)* '}')?
(AddedTags)?;

La especificación del elemento *Kernel* incluye un ID y una descripción obligatoria del mismo. Adicionalmente, puede contener expresiones que lo definen como:

'with rules' es una descripción que contiene las posibles reglas de ese elemento.

'owns' en esta expresión se especifican los elementos (cero o muchos) que contiene el elemento *Kernel* que se está definiendo. Estos elementos pertenecen al grupo *KernelElement.*

'uses' expresión en la que se definen los posibles elementos (cero o muchos) que se relacionan con el elemento *kernel* que se está definiendo y estos elementos deben pertenecer al grupo *KernelElement*. El usar la expresión *KernelElementRef* implica que contiene el ID de los posibles (cero o muchos) *KernelElement.*

Practice:
'practice' ID ':' STRING
'with objective' STRING
('with measures' STRING(',' STRING)*)?
('with entry' STRING(',' STRING)*)?
('with result' STRING(',' STRING)*)?
('with rules' STRING)?
('owns' '{' PracticeElement* '}')?
('uses' '{' PracticeContentRef (','PracticeContentRef)*'}')?
(AddedTags)?;

La especificación del elemento *Practice* incluye un ID, una descripción y un objetivo obligatorios. Adicionalmente, puede contener expresiones que lo definen como:

'with measures' es una descripción que contiene las posibles medidas de ese elemento.

'with entry' es una descripción que contiene las posibles entradas de ese elemento.

'with result' es una descripción que contiene los posibles resultados de ese elemento.

'with rules' es una descripción que contiene las posibles reglas de ese elemento.

'owns' en esta expresión se especifican los elementos (cero o muchos) que contiene el elemento *Practice* que se está definiendo. Estos elementos pertenecen al grupo *PracticeElement.*

'uses' expresión en la que se definen los posibles elementos (cero o muchos) que se relacionan con el elemento *Practice* que se está definiendo y estos elementos deben pertenecer al grupo *PracticeContent*. El usar la expresión *PracticeContentRef* implica que contiene el ID de los posibles (cero o muchos) *PracticeContent.*

C. *Practice Elements*

Activity:
'activity' ID ':' STRING
(Resource(',' Resource)*)?
'targets' StateOrLevelRef (',' StateOrLevelRef)*
('with actions' Action (',' Action)*)?
('requires competency level' CompetencyLevelRef(',' CompetencyLevelRef)*)?
(AddedTags)?;

La especificación del elemento *Practice* incluye un ID, una descripción y el cumplimiento de al menos un objetivo (grupo *StateOrLevel*). Adicionalmente, puede contener expresiones que lo definen como:

Resource es una expresión que contiene una fuente.

'with actions' es una expresión que muestra las acciones que se tiene sobre alfas o productos de trabajo.

'requires competency level' es una expresión que muestra el ID de las competencias necesarias para la actividad que se está describiendo.

AddedTags es información adicional para describir la actividad.

D. *Auxiliary Elements*

Resource:
'resource' (UserDefinedTypeRef '=')? STRING;

Es una expresión que contiene una fuente o información de donde se puede conseguir información.

IV. Representación

Para este artículo se realiza la declaración del lenguaje textual de Semat de tal manera que el lenguaje textual sea consistente con el lenguaje gráfico. A continuación se profundiza sobre los elementos modificados:

A. *Resource*

Este elemento se cambió ya que la expresión *Resource* necesita sólo un STRING para la descripción de la fuente del elemento.

Resource:
'resource' ID ':' STRING;

Lenguaje gráfico:
No tiene

Conexión con otros elementos: **UserDefinedType**.

B. *Kernel, Practice y Activity*

Estos elementos se modificaron para que las representaciones realizadas con estas declaraciones no tengan asociaciones con elementos que no existen en el lenguaje gráfico.

Kernel

Dentro de la definición de este elemento se especifica que se compone y relaciona con *KernelElement*. Sin embargo en el lenguaje gráfico, el *Kernel* no tiene relaciones con otoros elementos.

Adicionalmente, en el lenguaje textual se encuentran inconsistencias. A partir de la definición de *Kernel* en BNF, se puede inferir que el *Kernel* tiene un *AlphaAssociation*. La inconsistencia surge cuando se conoce la definición de este último elemento, el cual sólo se usa entre Alphas.

AlphaAssociation:
Cardinality AlphaRef '--' STRING '-->' Cardinality AlphaRef (AddedTags)?;

Por lo tanto la especificación del elemento para lograr la consistencia es la siguiente:

Kernel:
'kernel' ID ':' STRING
('with rules' STRING)?
(AddedTags)?;

De esta manera solo se espera un ID obligatoriamente al definir un *Kernel* y puede o no tener una descripción con sus reglas y algunos comentarios adicionales sobre el elemento. De esta manera, esta especificación es consistente con lo que muestra el lenguaje gráfico (véase la fig. 11).

Lenguaje gráfico:

Conexión con otros elementos: Ninguna

Figura 11. Conexiones del Kernel en el lenguaje gráfico [6]

Practice

Dentro de la definición de este elemento se especifica que se compone de *PracticeElement* y relaciona con *PracticeContent*. Sin embargo, en el lenguaje gráfico, *Practice* no tiene relaciones con otros elementos.

Adicionalmente, en el lenguaje textual se encuentran inconsistencias. A partir de la definición de *Practice* en BNF, se puede inferir que *Practice* tiene *PracticeElement*. A su vez *PracticeElement* puede ser *PatternElement* y éste último puede ser *ActivityAssociation*. La inconsistencia surge cuando se conoce la definición de este último elemento, el cual sólo se usa entre *Activities*.

Adicionalmente, la definición de *PatternElement* no resulta ser congruente con el lenguaje gráfico al tener asociada una *ActivityAssociation*.

ActivityAssociation:
AbstractActivityRef '--' STRING '-->' AbstractActivityRef (AddedTags)?;

Por lo tanto la especificación del elemento para lograr la consistencia es la siguiente:

Practice:
'practice' ID ':' STRING
'with objective' STRING
('with measures' STRING(',' STRING)*)?
('with entry' STRING(',' STRING)*)?
('with result' STRING(',' STRING)*)?
('with rules' STRING)?
(AddedTags)?;

De esta manera, sólo se espera un ID y un objetivo obligatorios al definir *Practice* y puede o no tener una descripción con sus medidas, entradas, resultados y algunos comentarios adicionales sobre el elemento. Así, esta especificación es consistente con lo que muestra el lenguaje gráfico (véase la fig. 12).

Lenguaje gráfico:

Conexión con otros elementos: Ninguna

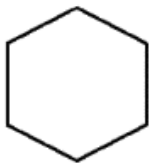

Figura 12. Conexiones de *Practice* en el lenguaje gráfico [3]

Activity

Dentro de la definición de este elemento se especifica que se compone de *StateOrLevelRef* o *CompetencyLevel.* Sin embargo, en el lenguaje gráfico, *Activity* no tiene relaciones con otros elementos.

Por lo tanto la especificación del elemento para lograr la consistencia es la siguiente:

Activity:
'activity' ID ':' STRING
(Resource(',' Resource)*)?
(AddedTags)?;

Por ello, sólo se espera un ID obligatorio al definir *Activity* y puede o no tener una descripción con la fuente y algunos comentarios adicionales sobre el elemento.

Las actividades, en consecuencia, se crean de forma independiente y para crear las diferentes asociaciones que muestra el lenguaje gráfico se utiliza el *ActivityAssociation*.

ActivityAssociation:
AbstractActivityRef '--' STRING '-->' AbstractActivityRef (AddedTags)?;

De esta manera esta especificación es consistente con lo que muestra el lenguaje gráfico (véanse las Fig. 13 y 14).

Lenguaje gráfico:

Conexión con otros elementos: **Activity** y **Activity Space**.

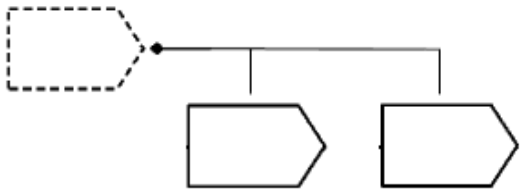

Figura 13. Conexión de **Activity** con **ActivitySpace** en el lenguaje gráfico [3]

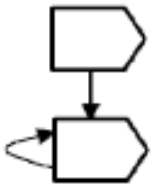

Figura 14. Conexión de **Activity** con **Activity** en el lenguaje gráfico [3]

C. *PatternElement*

En el lenguaje gráfico de Semat, el elemento *Pattern* tiene una asociación con cualquier elemento. Sin embargo, en el lenguaje textual aparecen más elementos auxiliares con los cuales el elemento *Pattern* no debería tener asociación, por ejemplo: **AlphaAssociation**, **AlphaContainment** y **ActivityAssociation**.

Por lo tanto, este elemento se editó con el fin de que los elementos que utilizan este objeto no puedan hacer referencia a las asociaciones entre los elementos que antes se presentaban.

PatternElement:
Alpha | WorkProduct | WorkProductManifest | Activity | ActivitySpace | Competency | Pattern;

De esta manera, esta especificación es consistente con lo que muestra el lenguaje gráfico (véase la fig. 15).

Lenguaje gráfico:

Conexión con otros elementos: Todos.

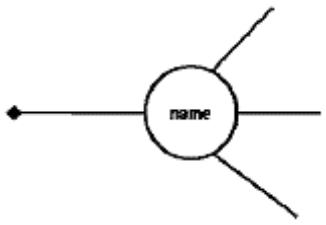

Figura 15. Conexión del **PatternElement** [3]

V. CONCLUSIONES

Las definiciones de un lenguaje dependen de tres componentes: semántica, sintaxis y pragmática. La semántica revela el significado de las frases que son correctamente formadas, la pragmática estudia los factores extralingüísticos del acto comunicativo que permiten entender, inferir o interpretar lo que el emisor de la frase quiso dar a entender y la sintaxis define la relación formal entre los constituyentes de un lenguaje, es decir, se refiere a la forma correcta de formar las frases en el lenguaje. Una definición formal de la sintaxis se denomina gramática, la cual se utiliza para especificar de manera finita el conjunto de cadenas de símbolos que constituyen un lenguaje.

La forma BNF es una manera explícita de representar la gramática libre de contexto y se utiliza en lenguajes de programación para hacer y expresar la estructura de manera formal mediante símbolos. EBNF permite el uso de algunas reglas adicionales que facilitan la especificación y completitud del lenguaje.

Aunque la declaración de Semat en el lenguaje EBNF tiene una sintaxis correcta, la semántica muestra un lenguaje diferente al lenguaje gráfico. Semat cuenta con dos tipos de lenguajes, el lenguaje gráfico y el lenguaje textual, los cuales describen los diferentes elementos que componen el núcleo de esta iniciativa. Sin embargo, los lenguajes aun presentan falencias, pues se encontraron inconsistencias en la especificación de los elementos. Los elementos auxiliares que define el núcleo de Semat en EBNF permiten la definición de otros elementos del núcleo.

En el lenguaje textual de Semat se especifican elementos auxiliares que son necesarios para poder representar las asociaciones entre los elementos del núcleo de Semat. Sin embargo, una especificación incorrecta de estos elementos puede representar asociaciones incorrectas, por ejemplo *PatternElement* el cual abarca elementos como *AlphaAssociation* y *ActivityAssociation* los cuales no son parte del elemento *Pattern*.

Siendo *Essence* una especificación aún en construcción en el OMG, se espera que sigan surgiendo versiones que contengan inconsistencias como las que se analizaron en este artículo. Por ello, el trabajo futuro se basa en la continuación del análisis de los elementos textuales de la especificación y su adaptación al lenguaje gráfico que describe Semat. Otra línea de trabajo futuro que da continuidad a este artículo se relaciona con otros elementos de la especificación de *Essence* que, aún sin declarar sus relaciones en el lenguaje textual o en el gráfico, permiten conexiones de tipo pragmático por medio de diagramas descriptivos o por uso de sus elementos.

AGRADECIMIENTOS

El proyecto identificado con el código 18907 y que lleva por título “Especificación formal en OCL de reglas de consistencia entre métodos de desarrollo basados en planes, representado en el núcleo de SEMAT”, que financia la Dirección de Investigación de la Sede Medellín (DIME), adscrita a la Universidad Nacional de Colombia suministró los fondos para la realización de este artículo.

REFERENCIAS

[1] I. Jacobson, P. P. W. Ng, P. E. McMahon, I. Spence, S. Lidman, C. M. Zapata (traductor), “La esencia de la ingeniería de software: El núcleo de Semat”, *Revista Latinoamericana de Ingeniería de Software*, vol. 3, pp. 71–78, 2013.

[2] N. Chomsky, “The independence of grammar”, en *Syntactics structures*, S. Wendland, Walter de Gruyter GmnH & Co. KG, Berlín, 1957, pp. 117.

[3] Essence – *Kernel and Language for Software Engineering Methods. Versión 1.3*, 2013.

[4] B. L. Kurtz, K. Slonneger, ”Specifying syntax”, en *Formal syntax and semantics of programming languages. A laboratory based approach*, 1995, pp. 625.

[5] D. E. Knuth, “Backus Normal Form vs. Bakus Naur Form”, *Communications of the ACM*, vol. 7, no. 12, 1964, pp. 735–736.

[6] L. Reynoso, M. Genero, M. Piattini, “Towards a metric suite for OCL Expressions expressed within UML/OCL models”, *Journal of Computer Science & Technology*, vol. 4, no.1, 2004, pp. 38.