Una Componente de Streams para Contabilizar la Métrica Web Número de Enlaces Rotos

Riesco, Daniel[1]
Berón, Mario Marcelo[1]
Montejano, Germán[1]
[1] *Project Software Engineering Method and Tool.*
Universidad Nacional de San Luis, Argentina
Address: ***{driesco|mberon}@unsl.edu.ar***

Dosch, Walter[2]
[2]*Institute of Software Technology and Programming Languages*
University of Lübeck, Germany
Address: ***dosch@isp.uni-luebeck.de***

RESUMEN

Un stream es una secuencia finita o infinita de mensajes transmitidos por un canal. El formalismo de stream define un conjunto de reglas y propiedades. Este formalismo hace posible modelar cierto dominio de aplicación por medio de la especificación de componentes de software.

Esta línea de investigación estudia el uso del *Formalismo de Streams* para la definición de *Métricas Web*. A modo de ejemplo, se presenta la especificación formal de la métrica Web: *Número de Enlaces Rotos*. El trabajo consiste en tener una componente de streams que reciba como entrada páginas Web, y produzca como resultado una cadena de enteros que indique el número de enlaces rotos por cada una ellas.

Para construir esta componente se definen tres transformadores de streams básicos *Encontrar*, *Verificar*, *Sumar*. *Encontrar,* selecciona los enlaces de una página Web. *Verificar,* recibe como entrada un *stream de enlaces* y produce como salida un *stream de enlaces rotos*. *Sumar*, produce un *stream de enteros*, que indica la cantidad de enlaces rotos por cada una de las páginas Web analizadas.

Estos transformadores de streams son combinados para producir la componente que contabiliza los enlaces rotos de una secuencia de páginas Web.

Palabras Claves: Streams, Métricas Web, Componentes, Número de Enlaces Rotos.

1. INTRODUCCIÓN

La Internet ha experimentado un gran crecimiento en las últimas décadas debido a la amplia variedad de servicios que provee. Este hecho, condujo al estudio de las características que deben poseer los programas que se usan en la Web para que el usuario pueda llevar a cabo las tareas con efectividad, seguridad y satisfacción[5,7].

La clase predominante de software en la Internet está formada por páginas Web. Desde ahora para lograr un mejor servicio y calidad en la red de redes, es necesario que este software satisfaga las necesidades de los usuarios[8]. Una forma de analizar el grado de satisfacción de los usuarios de esta clase de productos, es proveer un mecanismo de evaluación de la calidad de este tipo de software[4,6].

La evaluación del software Web no es una tarea sencilla ya que es difícil tener en cuenta todas las características, atributos deseables y obligatorios del mismo. Esta observación conduce a la consideración de metodologías que ayuden a evaluar los productos Web. Una de ellas es la metodología QEM (Quality Evaluation Model), cuyo modelo de calidad se basa en el estándar ISO 9126 que declara:

"La calidad de un producto queda definida a un alto nivel de abstracción por las características denominadas usuabilidad, funcionalidad, confiabilidad, eficiencia, mantenibilidad y portabilidad."

Dentro de la característica confiabilidad se encuentra definido el atributo número de enlaces rotos. Dicho atributo hace referencia a aquellos enlaces que el usuario utiliza y no pueden ser recuperados. La presencia un número significativo de este tipo de enlaces afectan en gran medida la calidad del software.

El chequeo de esta métrica es un problema que puede ser especificado utilizando el formalismo de streams[9,10,13]: cada máquina que desea acceder a una página web recibe como entrada un stream de código html. De esta forma se define una componente que recupere los enlaces de la página y verifique si su destino no puede ser alcanzado. Los enlaces pueden ser detectados porque siguen un patrón específico en su código html.

En este articulo se presenta la forma de definir formalmente métricas usando el formalismo de streams[12], en particular se define la componente que contabiliza los enlaces rotos de una secuencia de páginas web. Se pretende, en el futuro, traducir las especificaciones realizadas con streams al lenguaje RSL (RAISE Specification Language)[2] del método RAISE[1] (Rigorous Approach to Industrial Software Engineering), con el fin de realizar chequeos automáticos a las especificaciones[3] .

2. ENLACES EN EL LENGUAJE HTML

El objetivo de esta sección es introducir brevemente al lector en la especificación de enlaces en el lenguaje html. Esto facilitará el entendimiento de las especificaciones de los transformadores de streams necesarios para la construcción de la componente *Número de Enlaces Rotos.*

La definición de un enlace en el lenguaje html se lleva a cabo por la directiva <a>. Esta directiva debe incluir el parámetro href = "URL" para especificar el destino del enlace. En otras palabras, se debe escribir y finalizar con . Por ejemplo, si se desea que el texto "presione aquí para visitar a UNSL" conduzca a la página principal de la Universidad Nacional de San Luis, se debe escribir el siguiente código html:

Presione Aquí para visitar a la UNSL

Por otra parte, si se necesita hacer referencia a una ubicación que se encuentra en el mismo servidor o en un subdirectorio del mismo, es suficiente dar el nombre del archivo y opcionalmente el paso. Un ejemplo de esto es el siguiente:

 Presione Aquí

En las siguientes secciones se usará esta información para definir los transformadores de streams que conforman a la componente *Número de Enlaces Rotos*.

3. EL TRANSFORMADOR DE STREAM ENCONTRAR ENLACES

El transformador de stream *Encontrar* recibe como entrada un stream de código html correspondiente a las páginas Web que se desean analizar. *Encontrar*, recupera los enlaces de cada página y los incorpora en el stream de salida. La visión de caja negra del transformador se puede ver en la figura 1.

Figura 1: Visión de caja negra del transformador de stream Encontrar

El dominio de *Encontrar* esta formado por las páginas Web. Éstas pueden ser expresadas como sigue:

- D= Palabras claves del lenguaje html.
- ID= Texto definido por el usuario.

- Página = (D U ID)*
- Páginas = Página*

El rango de *Encontrar* esta formado por los enlaces encontrados en cada una de las páginas analizadas, simbólicamente:

- Enlaces_Por_Página = (Páginas – D)*
- Enlaces = Enlaces_Por_Páginas*

El transformador de streams *Encontrar*, hace uso de la función auxiliar hd3tl cuya especificación es la siguiente:

$$hd3tl: D \rightarrow D$$
$$hd3tl = (hd \circ tl \circ tl)(S)$$

Especificación 1: Función hd3tl

su finalidad es extraer el nombre del enlace.

Luego de haber definido el dominio y rango del transformador *Encontrar* y las funciones auxiliares necesarias, se puede especificar el comportamiento del mismo como sigue:

$$\text{Encontrar: } D \times \text{Páginas} \rightarrow \text{Enlaces } U \{\perp\}$$
$$\text{Encontrar}(e,S) = \begin{cases} <> & \text{si } S=<> \\ hd3tl(S) < \text{Encontrar}(e, tl(S)) & \text{si } hd(S)=e \\ \text{Encontrar}(e,tl(S)) & \text{en o.c.} \end{cases}$$

Especificación 2: Transformador de Streams Encontrar

4. EL TRANSFORMADOR DE STREAMS VERIFICAR

El objetivo del transformador de streams *Verificar* es comprobar que los enlaces encontrados puedan ser recuperados.

La visión de caja negra de este transformador es la siguiente:

Figura 2: Visión de caja negra del transformador de streams Verificar

Verificar, tiene como dominio un stream de streams de enlaces, denotado por E, y una función binaria f. El rango del transformador esta formado por un stream de streams de ceros y unos, donde cero indica que el enlace no pudo ser recuperado, y uno el caso contrario.

La especificación de la componente *Verificar* en el formalismo de streams se muestra en la especificación 3:

$$\text{Verificar} : E \times [E \times S \rightarrow E] \rightarrow E^* \rightarrow S^*$$
$$\text{Verificar}(e,f)(S) = \text{scan}(e,f)(S)$$

Especificación 3: Transformador de streams Verificar

la especificación de la componente scan puede ser vista en[11].

La función f usada por *Verificar* se especifica como sigue:

$$f: s \times E \rightarrow (\{0,1\}^*)^*$$
$$f(s,e): \begin{cases} <0> & \text{si } s=<> \\ 1 < f(tl(s),e) & \text{si } hd(s) \in e \\ 0 < f(tl(s),e) & \text{si } hd(s) \notin e \end{cases}$$

Especificación 4: Especificación de la función f

5. EL TRANSFORMADOR DE STREAM SUMADOR

El objetivo del transformador de streams sumador es contabilizar los enlaces rotos de las páginas Web. La visión de caja negra de *Sumador* es la siguiente:

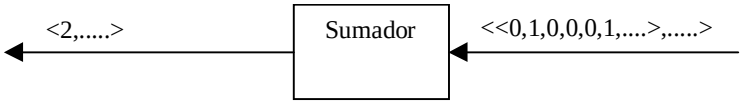

Figura 3: Visión de Caja Negra del Tramsformador de Streams Sumador

Sumador recibe como parámetro un stream de streams de ceros y unos denotado por ({0,1}*)*, y produce como resultado un stream de naturales que indica la cantidad enlaces rotos encontrados en las páginas. Su especificación se muestra en la especificación 5:

Sumador: $(\{0,1\}^*)^* \rightarrow N^*$
Sumador(S)= (last °scan(0,f))(S)

Especificación 5: Transformador de Streams Sumador

El transformador de streams last retorna el último elemento de un stream y f se define como sigue:

$$f: N \times \{0,1\}^* \rightarrow N$$

$$f(e,s): \begin{cases} <> & \text{si } s=<> \\ (e+hd(s)) < f((e+hd(s)),tl(s)) & \text{en o.c.} \end{cases}$$

Especificación 6: Función f, que implementa la suma de dos números

6. COMPONENTE CONTABILIZAR ENLACES ROTOS

En esta sección se presenta la especificación de la componente *Número Enlaces Rotos*, la visión de caja negra se muestra en la figura 4:

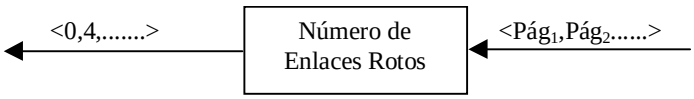

Figura 4: Visión de caja negra de la componente Contabilizar Enlaces Rotos

y la visión interna puede ser vista en la figura 5.

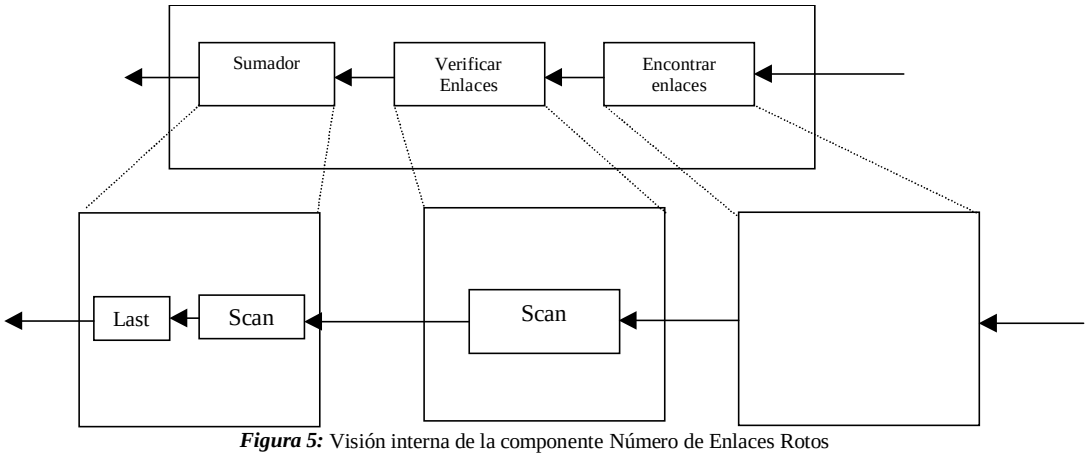

Figura 5: Visión interna de la componente Número de Enlaces Rotos

El comportamiento de la componente *Número de Enlaces Rotos* está dado por el siguiente transformador de streams:

Página = html* ; Páginas = Página* ; Patrón = html
Contabilizar_Enlaces_Rotos: Patrón x Páginas → Z*

$$\text{Contabilizar\_Enlaces\_Rotos}(p,S) = \begin{cases} <> & \text{si } S=<> \\ (\text{sumador}\circ\text{Check}(e,f)\circ\text{Encontar}(p,S))(hd(S)) < \\ \quad \text{Contabilizar\_Enlaces\_Rotos}(p,tl(S)) & \text{si } S\neq<> \end{cases}$$

Especificación 7: Componente Contabilizar Enlaces Rotos

7. CONCLUSIONES Y VISIÓN DE FUTURO

En este artículo se presentó una forma de utilizar el formalismo de streams para especificar software Web. Se observó que el formalismo de streams presenta las ventajas de: poder especificar nuevas componentes de software por medio de la reutilización de otras definidas previamente; analizar el comportamiento de las historias de entradas y salidas de cada una de las componentes software y además posee primitivas y reglas generales que permiten especificar software desde distintas visiones, lo cual es una ayuda para el entendimiento de la componente que se está especificando.

Siguiendo esta línea de investigación, se espera poder especificar componentes de software Web lo suficientemente generales de forma tal que tengan una amplia utilización en la construcción de nuevas componentes. Además se pretende utilizar RSL (RAISE Specification Language) como lenguaje formal para la especificación de esta clase de componentes de manera de lograr la definición precisa de la especificación con la consecuente realización de chequeos automáticos y prototipación en el lenguaje SML (Standard Mathematical Language) y la definición de axiomas y teoremas que amplíen la teoría de streams.

8. REFERENCIAS

1. *The RAISE Development Method*. Chris, George; Haxthausen, Anne E.; Hughes, Steven; Milne, Robert; Perhn, Soren; Pedersen, Jan Storbank. Prentice Hall. 1995.
2. *The RAISE Specification Language*. Chris, George; Haxthausen, Anne E.; Hughes, Steven; Milne, Robert; Perhn, Soren; Pedersen, Jan Storbank. Prentice Hall. 1992.
3. RAISE tools. www.iist.unu.edu.
4. *Hacia la Medición de Calidad en Uso Web*. Gonzalez Rodriguez, Julia; Olsina, Luis. 2001.
5. *E-Commerce Site Evaluation: a Case Study*. Olsina, Luis; Lafuente, Guillermo; Rossi, Gustavo. 2001.
6. *Catalogando Métricas Web*. Lafuente, Guillermo; Olsina, Luis. 2001.
7. *Web-site Quality Evaluation Method: a Case Study on Museums*. Olsina Santos, Luis. 2002
8. *Métricas de Calidad centradas en accesibilidad para KAI*. González, J.; Macías, A.; Nieto M. A.; Sánchez, F.. 2001.
9. *The semantics of a Simple Language for Parallel Programing*. 471-475. Kahn, G.; Information Processing. J. L. Rosenfeld. Amsterdam: North-Holland. 1974.
10. *A Survey of Stream Processing*. Stephens, R.. Acta Informática 34, 491-594; 1997.
11. *Scanning Stream*. Dosch, Walter. Institute of Software and Programming Languages. Medical University o Lübeck. Proceeding of the ACIS. International Conference on Computer Science, Software Engineering, Information Technology, e-Business, and Application 61-67. Foz do Iguazu. 2002.
12. *Views of a Bounded Stack*. Dosch, Walter. Institute of Software and Programming Languages. Medical University o Lübeck. Proceeding of the ACIS. International Conference on Computer Science, Software Engineering, Information Technology, e-Business, and Application 11-17. Foz do Iguazu.2002.
13. *Coroutines and Networks of Parallel Processes*. 993-998a. Information Processing; B. Gilchrist. Amsterdam: North-Holland. 1977.