ICT-UNPA-64-2013
ISSN: 1852 – 4516
Aprobado por Resolución Nro. 0579/13-R-UNPA

EXPLORACION DE REPOSITORIOS DE SOFTWARE Y ANALISIS DE POTENCIALES EXTENSIONES A ASPECTOS

Lic. Graciela Beatriz Vidal
vidalgracielaunpa@gmail.com
GISP- Instituto de Tecnología Aplicada
Universidad Nacional de la Patagonia Austral
UARG
2012

Resumen. Tanto el Desarrollo basado en Componentes como el Desarrollo de Software orientado a Aspectos son enfoque propuestos para abordar diversos problemas que surgen en el proceso de desarrollo de aplicaciones. Ambos enfoques tienen como objetivo el reuso de software, ante esta situación surge la necesidad de que componentes y aspectos compartan un espacio común en el cual puedan ser publicados, recuperados y reutilizados. Los repositorios actuales parecen no proveer aspectos de manera explícita, en ocasiones los aspectos son utilizados para especificar el componente, pero no son reusables. La tarea de publicar y recuperar componentes es muy compleja, existen diversos métodos de publicación y recuperación a tal fin. En este trabajo se exploraron diferentes repositorios analizando criterios al igual que se exponen los métodos de publicación y recuperación con el fin de determinar si es posible la adaptación o extensión de los repositorios a aspectos.

Palabras Clave: Componentes software, Repositorio de Componentes, Métodos de Publicación y Recuperación, aspectos.

1. INTRODUCCION

La Ingeniería de Software proporciona diferentes metodologías de desarrollo, las mismas pueden variar de acuerdo a la naturaleza de las aplicaciones, requerimientos de clientes, nuevas tecnologías, nuevos paradigmas, etc. Los requerimientos de usuarios cada vez más exigentes, como por ejemplo, el desarrollo rápido, aplicaciones de calidad, seguridad, etc. han ocasionado problemas en el proceso de desarrollo de aplicaciones. Con el fin de abordar algunos de estos requerimientos se ha incrementado el desarrollo orientado al reuso, el cual consiste en la reutilización de Aplicaciones Completas, Componentes o bien Objetos y Funciones [1].

El Desarrollo basado en el reuso de Componentes (DBC) [2][3] es una metodología muy utilizada en la actualidad dando lugar a la creación de Repositorios de Componentes (RC), posibilitando su publicación, recuperación y reuso.

Por otro lado el Desarrollo de Software Orientada a Aspectos (DSOA) [4][5][6], ha sido propuesto para brindar el soporte necesario para la separación de requerimientos no funcionales o crosscutting concerns[7]. Los crosscutting concerns, como por ejemplo logging, sincronización, replicación, entre otros, entrecruzan la funcionalidad principal de un sistema, generando el denominado código disperso y enmarañado, el cual presenta varias dificultades, como por ejemplo, el mantenimiento. Algunas de las ventajas del DSOA son lograr aplicaciones más flexibles, adaptables como así también el reuso de los componentes utilizados en el sistema. Un *aspecto* es la abstracción central del DSOA, el cual se compone de dos partes modulares: *pointcuts* y *advices*. Los advices son similares a los métodos, debido a que son fragmentos de código que serán incorporados al dominio en algún momento (*after, before, around*); los pointcuts son las expresiones que establecen ante qué eventos y condiciones estos fragmentos se ejecutarán, por ejemplo la llamada a un determinado método [4].

Un componente debe ser claramente encapsulado en un procedimiento generalizado, es decir, bien localizado, fácilmente accedido y compuesto cuando sea necesario[8].

Ante las metodologías mencionadas, que presentan objetivos en común, surge la idea de unificar aspectos y componentes en un solo ambiente, más precisamente en un repositorio para su posterior reuso, lo que motiva al análisis de los mismos. En primer lugar, si actualmente los repositorios proveen aspectos reutilizables, de comprobarse que no es así, el objetivo es realizar adaptaciones o extenderlos de manera que aspectos y componentes puedan ser publicados y recuperados de un ambiente común.

El objetivo de este trabajo es explorar y analizar repositorios de software actuales, como así también los métodos de publicación y recuperación utilizados, debido a que los mismos juegan un papel fundamental para adaptar o extender los repositorios a un ambiente de DSOA.

Como primer paso en esta investigación, se llevó a cabo una Revisión Sistemática de la Literartura (RSL)[9] a fin de definir un marco de trabajo que permita seleccionar información relevante ante la gran cantidad de documentación a la cual se puede acceder actualmente.

El presente trabajo esta organizado de la siguiente manera: en la Sección 2 se expone la metodología utilizada para llevar a cabo este trabajo, en la Sección 3 se definen Repositorio de Software y especificación de componentes, en la Sección 4 se desarrollan los Métodos de Publicación y Recuperación existentes, en la Sección 5 se presentan características generales sobre repositorios disponibles, en la Sección 6 se exponen y definen criterios que luego son analizados sobre cada uno de los repositorios, finalmente en la Sección 7 se presentan las conclusiones.

2. REVISION SISTEMATICA DE LA LITERATURA

La variada y extensa cantidad de material disponible relacionada a los temas a desarrollar dio origen a la necesidad de llevar a cabo una Revisión Sistemática de la Literatura (RSL). La misma define una manera de evaluar e interpretar toda la investigación disponible, que sea relevante respecto de una interrogante de investigación particular, un área temática o fenómeno de interés[9]. Una RSL proporciona un marco de trabajo en el cual es posible realizar búsquedas detalladas en lugares específicos, seleccionar documentos con alto valor científico, evaluar y sintetizar la información existente relacionada al tema de interés.
Se definieron protocolos de búsqueda y revisión, los que debieron ser ajustados, debido a que eran demasiado genéricos, por lo tanto los resultados obtenidos incluían información que no era de interés para el trabajo.

Protocolo de búsqueda

El primer paso de la RSL fue la búsqueda en internet, los sitios utilizados a tal fin fueron:

- http://scholar. google.es[1]
- http://www.scielo.org.ar/scielo.php[2]
- http://www.scirus.com[3]

Se definieron combinaciones de palabras clave para realizar búsquedas en sitios web científicos de manera que se pueda acceder a textos con cierto reconocimiento en el área a investigar, lo que otorga mayor fiabilidad en la misma.
Luego se identificaron palabras clave asociadas a los temas de investigación y combinaciones de las mismas, tales como:

4 *Ingeniería de software.*
- *Reuso.*
- *Componentes.*
- *Repositorios.*
- *Desarrollo basado en el reuso.*
- *Programación orientada a aspectos.*
- *Ingeniería de Software + reuso*
- *Ingeniería de Software + componentes*
- *Ingeniería de Software + Desarrollo basada en reuso de Componentes*
- *Repositorio + Programación orientada a Aspectos*
- *Repositorio + POA,etc*

Además de estas combinaciones se realizaron búsquedas utilizando estos términos en inglés. Teniendo en cuenta que las siglas son muy utilizadas, se realizaron búsquedas con las siguientes:

Ingeniería de Software basada en Componentes (ISBC).
Programación Orientada a Aspectos (POA), AOP en inglés.
Component-based software development (CBD).
Desarrollo de Software Orientado a Aspectos (DSOA), AOSD en inglés.

---

1 http://scholar. google.es

2 http://www.scielo.org.ar/scielo.php

3 http://www.scielo.org.ar/scielo.php

Se definieron criterios de selección sobre la información obtenida como resultado de las búsquedas definidas:

- Documentos que fueron presentados o publicados recientemente en eventos de reconocidos, conferencias, workshop, congresos, etc.
- Documentos de autores reconocidos en el área de investigación.
- Documentos que cumplan correctamente con los formatos establecidos, que cuenten con referencias comprobables, que
- Documentos que presenten tablas de resumen, gráficos, esquemas.
- Documentos que hagan referencia a trabajos relacionados y/o trabajos futuros.
- Documentos que contengan la mayor cantidad de información sobre sus autores.(Institución, correo electrónico, formación académica, algunos antecedentes, etc.)

Con estos resultados fue necesario refinar las búsquedas, debido a que arrojó información poco relevante, en algunos casos no relacionada con el tema de investigación.

En algunos sitios se observan gran cantidad de resultados para algunas combinaciones de palabras, sin embargo estas búsquedas fueron descartadas por contener información que no relevante con el trabajo en estudio.

Observación

Como primer criterio para seleccionar los estudios primarios, se verificó que cumpliera con las características definidas en el protocolo de búsqueda, tales como: autor, tipo de documento, cantidad de páginas, año de realización, entre otros.

La cantidad de material que cumple con estas restricciones es bastante, por lo que se tuvo en cuenta aquellos documentos en cuyo título aparezcan las palabras de búsqueda o su combinación. Los resultados de esta selección se resumen

Selección y síntesis de datos

Teniendo en cuenta las prioridades asignadas en la etapa anterior se lleva a cabo una lectura más profunda sobre aquellos documentos de mayor interés. Luego se dará inicio a la síntesis de los datos obtenidos, esta tarea consiste en documentar el desarrollo de los siguientes temas:

6.2 Desarrollo Basado en Componentes: definición, etapas, quién la desarrolla?

6.3 Desarrollo Basado en el Reuso: qué es? Sus ventajas y desventajas.

6.4 Repositorio de Componentes: en qué consiste? Qué tipo de componentes y/o artefactos contiene? Tipos de repositorios, si existen varios.

6.5 Publicación y recuperación de Componentes: en qué consiste?, quién la lleva a cabo?

6.6 Aspectos reutilizables: definición de aspecto, son reutilizables?, en qué medida? , depende del lenguaje de implementación?

Posteriormente se buscaron los repositorios propiamente dichos, con el fin de acceder a ellos, analizar sus características principales y los criterios que se expondrán en Sección x. En la actualidad existen gran cantidad de repositorios con diferentes fines, comerciales, académicos, investigación, etc., además todos ellos proveen diversos recursos a los cuales acceder y se enfocados a diferentes áreas, algunos muy específicos y otros más genéricos.

Como criterio se selección se tuvo en cuenta, que fueran reconocidos por los desarrolladores, que sean visiblemente atractivos en cuento a los recursos que proveen (cantidad y diversidad), que contaran con el apoyo o respaldo de entidades educativas, como Universidades. También se han seleccionado algunos de dominio específico con el fin de observar si su estructura y funcionamiento varía con respecto a uno de dominio específico.

3. REPOSITORIOS DE SOFTWARE

El reuso de software es el proceso por el cual una organización define un conjunto de procedimientos semánticos que le permitan especificar, producir, clasificar, recuperar y adoptar artefactos software, con el objetivo de utilizarlos en sus diferentes actividades[10].
A diferencia de otros enfoques, el DBC se enfoca en la integración de componentes software existentes. Para esto, se requiere de un conjunto de componentes software disponible que puedan ser reusados, lo que se denomina repositorio o librería de software, el mismo puede almacenar componentes propios, construidos para otras aplicaciones o componentes software adquiridos a terceros.

3.1 REPOSITORIOS

Los repositorios son fundamentales para un adecuado DBC, en él se almacenarán los componentes, que se encuentran disponibles para ser recuperados y reusados [11].
Un repositorio o librería de software debe estar diseñado para proveer un soporte tan automatizado como sea posible que permita al usuario identificar, comparar, evaluar y recuperar los componentes. Es deseable también, que provea soporte para adaptar, transformar y especializar componentes.
El DBC presenta algunos inconvenientes tales como: a) requerir la búsqueda y recuperación de un componente de software que coincida con las especificaciones de componentes, ya que los componentes se encuentran almacenados en repositorios con diferentes estructuras en la Web, b) seleccionar el más apropiado de una gran cantidad de componentes y c) integrar el componente a los ya desarrollados, previas adaptaciones si es necesario, para construir una nueva aplicación software de dominio específico[12][13].
Por estas razones el repositorio debe recibir mantenimiento, de manera que los componentes deseados sean recuperados cuando sea necesario. En este proceso es necesario proponer una clasificación de componentes, su navegación, técnicas de búsquedas y recuperación automatizadas[14]. La investigación en recuperación de componentes es altamente dinámica, se requieren repositorios efectivos, fáciles de usar, lo más complejos posibles en cuanto a la funcionalidad de los componentes que provee y que además sean efectivos en la recuperación de los mismos. Para que el DBC sea exitoso debe estar respaldado por una definición, especificación y contextualización de componentes. Los términos componentes, artefactos o activos no se refieren únicamente a código fuente o ejecutables, en un repositorio es posible encontrar especificaciones de diseño, diagramas, papers, tutoriales, etc. sin embargo en la actualidad no es posible recuperar y reusar módulos que encapsulen aspectos dentro de estos repositorios. Ante la necesidad de incluirlos, la especificación de estos módulos, es un punto a tener en cuenta, debido a que determinará en gran parte, el método de publicación y recuperación a ser utilizado.

Especificación de un Componente

Un componente es una unidad de composición con interfaces bien especificadas, diseñado para ser utilizados como plug y para ser interpretable[11], debe ser posible encapsularlo claramente en un procedimiento generalizado, es decir, debe estar bien localizado, fácilmente accedido y compuesto como sea necesario[15]. Se pueden describir diferentes formas de especificación de componentes para que sean fácilmente identificables e integrados a una estructura o jerarquía.

Inicialmente un componente puede describirse en términos de su interface, pero esto carece de información adicional del componente como: desempeño, seguridad y sincronización[16].
La especificación de un componente se enfoca en dos características, los metadatos y el comportamiento del mismo. Los metadatos de cada componente deben proporcionar la información necesaria para recuperarlo e implementarlo. Además, deben ayudar a los usuarios de repositorios a determinar si los artefactos son los adecuados y proveer información acerca de cómo utilizarlo cuando sean recuperados[17].
Sin embargo, los metadatos fallan en la representación de la funcionalidad de los componentes mas allá de las categorías generales y búsquedas basadas en texto libre. En la actualidad los repositorios tienden a capturar el comportamiento a través de palabras clave describiendo un área de funcionalidad general o como un campo de descripción en los metadatos[18].

4. METODOS DE PUBLICACION Y RECUPERACION DE COMPONENTES

Localizar e identificar componentes adecuados es uno de los mayores problemas en el reuso de componentes[19][20][21]. En los métodos de recuperación de un componente deben considerarse tres factores: la descripción del componente, especificación de las consultas de los usuarios y el proceso de recuperación. Además se considera que el reuso de componentes, supera los límites de una empresa u organización de desarrollo de software, es decir, en lugar de obtener los componentes de la librería de la empresa, los usuarios pueden buscar los componentes deseados en otros repositorios.
El rápido crecimiento en el número de componentes demanda una estricta eficiencia en la recuperación de componentes[21]. La mayoría de los métodos propuestos para solucionar los problemas de recuperación asumen que el usuario puede definir claramente sus consultas sobre componentes. Los modelos de recuperación de componentes mas conocido serán descriptos en esta sección.

4.1 METODOS BASADOS EN TEXTO LIBRE

El método de recuperación de componentes más popular es el Basado en Texto Libre [22][23] y proviene de la comunidad de recuperación de información. En este método los componentes se representan como documentos basados en texto libre, mientras que las consultas son realizadas utilizando palabras clave. El proceso de recuperación busca estas palabras en todos los documentos de descripción de componentes, aquellos que tengan mas coincidencias serán seleccionados.
Se ha propuesto el uso de diccionarios que contengan sinónimos y antónimos para extender las palabras claves y de esta manera obtener componentes más relevantes[24]. El conocimiento de dominio general también es utilizado para lograr esta extensión[21].
Una variante de estos métodos son los basados en búsqueda incremental de fácil uso, con una fuerte retroalimentación con el usuario. Con este propósito, una colección de componentes inicialmente no estructurada, es estructurada acorde a sus conceptos según lo determinado por análisis de conceptos formales[25]. A cada componente se le asigna un número arbitrario de palabras clave.
Para la recuperación, los usuarios especifican una consulta como un grupo de palabras clave, son seleccionando todos los componentes con al menos algunas de las palabras clave asignadas. Las

palabras claves se eligen desde una lista, presentando solo las significativas, cada elección de palabras claves refina la consulta y reduce los resultados un grupo no vacío de componentes seleccionados.

4.2 METODOS BASADOS EN RAZONAMIENTO

El Razonamiento basado en Casos (CBR) es denominado, método de resolución de problemas [26].La idea principal de CBR es que frente a nuevo caso (nuevo problema), se tiende a buscar en la memoria, problemas anteriores similares y a reusar las viejas soluciones para ayudar a solucionar el nuevo problema. El proceso CBR puede dividirse en cuatro fases; recuperar, reusar, revisar y conservar[26].
En la fase de recuperación, un nuevo caso, es decir un nuevo problema, es comparado con casos almacenados, aquellos que son similares a este caso, son recuperados. En CBR se adopta la coincidencia parcial, esto significa, la coincidencia de un grupo de características para obtener un mejor resultado, donde cada una tiene sus propios valores. Algunos métodos de CBR son denominados de pobres en conocimiento, debido a que solo consideran similitudes superficiales o sintácticas entre un nuevo caso y los almacenados, mientras que otros toman en cuenta similitudes semánticas y sintácticas al combinar conocimiento de dominio específico y general, este enfoque se denomina Razonamiento basado en Casos y Conocimiento Intensivo[27].

4.3 METODOS BASADOS EN RAZONAMIENTO DEL CONOCIMIENTO CONVERSACIONAL

El Razonamiento basado en el Conocimiento Conversacional (CCBR), es una forma interactiva de CBR, utiliza un diálogo inicial para guiar al usuario en el proceso de recuperación a través de una secuencia de preguntas y respuestas[28].En el proceso CBR tradicional, los usuarios proporcionan la descripción de un problema bien definido, basándose en esta descripción, el sistema CBR busca los casos apropiados.
Este método incluye seis partes; una base de conocimiento, un módulo de generación de nuevos casos, un módulo CBR de alto nivel de conocimiento, un módulo de visualización de componentes, un módulo de categorización y generación de preguntas y un módulo de muestra de preguntas.
La base de conocimiento almacena tanto, conocimiento específico como de dominio general de componentes. El módulo de generación de un nuevo caso puede establecerlo basándose en las consultas iniciales del usuario y sus posteriores respuestas a preguntas discriminativas. Dado un nuevo caso, el módulo de conocimiento intensivo calcula las similitudes entre el nuevo caso y los almacenados, retornando el componente cuyas similitudes superan un determinado límite (el cual es especificado inicialmente y que puede ser ajustado en posteriores ejecuciones del sistema). El módulo de visualización de componentes, muestra a los usuarios los componentes candidatos ordenados por sus similitudes. En el módulo de generación de preguntas y clasificación se identifican posibles preguntas y se utiliza un algoritmo[29] de obtención de información para clasificar posibles preguntas, conforme a cuánta información puede proporcionar si fue respondida.
Finalmente, el conocimiento general es utilizado para filtrar la salida de esas preguntas, si las respuestas pueden ser inferidas desde las consultas iniciales o preguntas previamente respondidas.

El módulo de visualización de preguntas selecciona las preguntas discriminativas para optimizar la búsqueda hacia una significativa respuesta[27].

4.4 METODOS BASADOS EN LA TAXONOMÍA DE LOS COMPONENTES

Una característica importante de un Repositorio es la naturaleza o el tipo de componentes, artefactos o activos que almacena. No se almacena únicamente código fuente, otros componentes como especificaciones de diseño, documentos de análisis, descripciones de diseño, diagramas de análisis y diseño en notaciones UML, datos de pruebas, manuales de usuario, interfaces de componentes, también pueden ser reusados. El formato y estructura de cada tipo de documento varía significativamente por lo tanto, el método de publicación y recuperación de componentes depende fuertemente de la naturaleza de los activos. Algunas librerías poseen restricciones, en el sentido que trabajan con tipos de activos simples, mientras que otras lo hacen con un amplio rango de activos[30].

El activo más común almacenado en librerías es el código fuente o código ejecutable que constituyen la base para el reuso de caja blanca y caja negra respectivamente. El nivel de estructuración del reuso de activos varía desde una estructura cruda de documentos textuales tales como test, documentos de análisis textual y especificaciones de requerimientos hasta el alto grado de formato estructurado de componentes como el código fuente. Esta naturaleza heterogénea de colecciones es el mayor problema y restringe el uso de un método común de publicación y recuperación[30].

El éxito de la reutilización está basado en la habilidad para identificar y recuperar componentes reusables utilizando el método adecuado. A nivel general los componentes se pueden clasificar en Activos Textuales y Activos No Textuales, esta categorización define criterios para el proceso de coincidencias. Por ejemplo, el código fuente o la especificación de requerimientos corresponden a Activos Textuales, en este caso, las consultas del usuario se basan en palabras clave y son comparadas contra la información contenida sobre los componentes existentes. Cabe mencionar que estos métodos no son muy eficientes en cuanto a precisión y recall[4] [30].

Los componentes ejecutables o imágenes son Activos No Textuales, para su almacenamiento y recuperación se requieren métodos que involucren la representación de los mismos, alguna abstracción textual estructurada o no estructurada y entonces comparar el contenido de la consulta contra estas abstracciones.

Asimismo, los Activos Textuales pueden ser identificados como altamente estructurados, parcialmente estructurados o no estructurados y en ejecutables y no ejecutables. Mientras que los Activos No Textuales pueden ser una caja negra ejecutable o caja negra no ejecutable, debido a que no se accede a su estructura. Esta clasificación da lugar a la utilización de los siguientes métodos:

Métodos de Recuperación de Información: utilizan el lenguaje natural, similar al basado en Texto Libre, es apropiado para Activos Textuales[22][23].

Métodos Descriptivos: cada activo es clasificado por un metadato llamado Term Space, la consulta debe incluir palabras claves que serán comparadas contra los Term Space de los activos del repositorio. Es aplicable a Activos Textuales y no Textuales, de todas maneras el resultado de

[4] La *precisión* es la relación entre los resultados correctos sobre los resultados obtenidos, mientras que *recall* es la relación entre los resultados correctos sobre los que deberían haberse obtenidos.

estos métodos depende de la terminología utilizada, la que debe ser de uso común entre usuarios y administradores del repositorio[31].

Métodos de Semántica Operacional: estos métodos identifican a los activos únicamente en base a su comportamiento[32][33] y a unas pocas entradas simples seleccionadas aleatoriamente. La entrada especificada en una consulta es comparada con la salida producida por los activos almacenados. Esto métodos son adecuados para componentes ejecutables.

Métodos de Semántica Denotativa: utilizan la representación semántica (prototipos o signatura) para hacer la comparación. Los componentes se recuperan comparando su signatura con una consulta determinada. Son apropiados para Activos Textuales y No Textuales[33].

Métodos Topológicos: utilizan una especificación a partir de las características del activo para iniciar la consulta. La especificación se hace en algún lenguaje de notación formal [33][34][35] como Z o notación CSP, etc. Aquellos activos cuyas características principales coincidan con las especificadas de la consulta, son recuperados.

Métodos Estructurales: utilizan un patrón estructural del componente[21], son utilizados particularmente para componentes como código fuente y otros en los cuales su estructura es visible y que pueden ser reusados previas modificaciones. No son aplicables a componentes ejecutables sin código fuente.

Métodos genéricos utilizando XML: El lenguaje de etiquetado extensible XML aparece como un estándar clave para la representación de documentos estructurados en web, además incorpora métodos para la representación de metadatos, permitiendo el desarrollo de una arquitectura de Recuperación de Información[22].

4.5 METODOS BASADOS EN ONTOLOGIAS

Las ontologías pueden utilizarse para la definición semántica de los componentes, esto incluye su comportamiento y características no funcionales. Dicha semántica abarca dos características, el dominio del componente y el conocimiento del componente software [36]. Aunque una ontología suele asociarse a la definición de un contexto (o vocabulario), en este caso nos referimos no solo al contexto, sino también a las condiciones y características internas del componente, las cuales deben ser especificadas de alguna forma.

Una ontología puede definirse como la representación formal de los términos básicos que comprenden el vocabulario de un dominio específico, las relaciones que forman las asociaciones entre términos y un conjunto de axiomas, que son la reglas y restricciones para combinar términos y relaciones para definir extensiones del vocabulario[37][38][39].

El principal objetivo es recolectar, representar y compartir conceptos (semántica) tanto de las consultas de los desarrolladores o usuarios como de las especificaciones de componentes software para la búsqueda ontológica y la recuperación de componentes utilizando la técnica de similitud semántica.

Para la recuperación de componentes reusables, es necesario revelar la semántica de los mismos para las consultas del desarrollador. Las características de un componente deben ser expresadas en un documento, esta información debe ser confiable y lo suficientemente completa para describir el componente.

4.5.1 Definiendo características de un componente

Las características seleccionadas pueden mejorar la eficiencia y precisión de búsqueda y recuperación al remover términos redundantes y no relevantes. Estas características son fáciles de especificar, pero no capturan completamente la semántica. La semántica no funcional de los componentes brinda apoyo al usuario para ordenar los resultados, tales como tamaño del código, complejidad del mismo o rendimiento.

Las categorías de metadatos son importantes en un contexto de desarrollo basado en componentes, ya que apoyarán al proceso de recuperación, algunas de las utilizadas son:

Ontología para metadatos del dominio de la aplicación[40] : describe el componente reusable a partir de la familia de aplicaciones y del dominio de la aplicación software de la cual se extrae el mismo.

Ontologías de desarrollo de software: describe el desarrollo de software, entidades y procesos.

Modelo de componentes: ontologías que definen propiedades centrales de un componente software[41] esto incluye:

- **Especificación sintáctica:** especifica la sintaxis para el uso de servicios de componentes (tecnologías como COM o JavaBeans), especifica la semántica de esos servicios y sus propiedades.
- **Metadatos de componentes semánticos:** es una extensión de la Especificación sintáctica, donde un estado modelo es asociado con cada interface y operaciones que tienen pre y post condiciones, además de las condiciones de los componentes dentro de la interface[41].
- **Otras:** características importantes que están relacionadas al reuso de componentes software tales como factores de calidad, lenguaje de programación, tamaño del componente, complejidad del código, modelo de diseño del software, licencia, etc.

Las ontologías pertenecen al enfoque de representación del conocimiento y apuntan a proveer un entendimiento compartido tanto por desarrolladores como por usuarios de las aplicaciones. Las ontologías describen el dominio de interés en forma tal que pueda ser procesada por computadoras.

La ontología del conocimiento en el proceso de recuperación de componentes tiene dos aspectos diferentes, uno relacionado con la evaluación de las consultas para recuperar conocimiento en lugar de recuperar la especificación del componente y el otro involucra el uso de la estructura de conocimiento ontológico para razonar y desplazarse sobre el dominio, el cual está recubierto por ésta base de conocimiento para la especificación de componentes.

La coincidencia semántica [42] es utilizada para realizar búsquedas, lo que permite encontrar relaciones semánticas más fuertes entre componentes objetos y consultas, dependiendo de una base semántica para recuperar componentes relevantes. Se calcula el grado de similitud, se toma en consideración si un concepto es parte de otro concepto, si es consistente con otro concepto, si es equivalente a otro concepto, o bien si es disjunto con otro concepto. Todo esto se logra utilizando reglas semánticas.

Las relaciones semánticas son identificadas en el proceso de minería de reglas, que utiliza el proceso de selección de características. Este proceso es aplicado a cada especificación de componentes ontológica.

Finalmente los componentes que coinciden con estas características son recuperados y presentados a los usuarios.

5. EXPLORACION DE REPOSITORIOS DE SOFTWARE

En secciones anteriores se presentaron conceptos teóricos, relacionados al DBC, Repositorios de Software y Métodos de publicación y recuperación de los mismos. Con la idea de extender estos repositorios a un entorno de DSOA, es necesario realizar una experiencia sobre los repositorios propiamente dichos.
Con este fin se accedió a diferentes repositorios disponibles en la actualidad, todos ellos son sitios web, con diferentes propósitos, algunos estrictamente comerciales, mientas que otros con fines académicos o de investigación, algunos de dominio específico sobre un área, mientras que otros ofrecen múltiples aplicaciones.
También se han consultado trabajos basados en repositorios que actualmente no se encuentran disponibles, pero que han sido de gran utilidad [43][44][45]. Se accedió a trabajos de investigación[46] en los cuales se utilizan los aspectos para indexar y describir componentes, basándose en sus características sistemáticas, incluyendo su interface de usuario, persistencia, distribución, seguridad, etc. Las consultas de los usuarios para recuperar componentes o solicitar servicios, son de alto nivel y están basadas en aspectos, pero éstos no son almacenados en los repositorios para ser reutilizados.
En esta sección se presentan brevemente características de cada uno de ellos, para luego analizar detenidamente algunos criterios y así definir cuáles pueden ser directamente aplicados al DSOA y cuáles requiere de modificaciones o adaptaciones, en este último caso de propondrán ideas iniciales para su concreción en trabajos futuros.

5.1 Apache UIMA

El proyecto Apache UIMA[5] (Unstructured Information Management Arquitecture) consiste en aplicaciones software que analizan grandes volúmenes de información no estructurada para descubrir conocimiento relevante para los usuarios finales.
Apache UIMA (Figura 1), provee una implementación open source con Licencia Apache , en el sitio es posible acceder a frameworks, componentes e infraestructuras. La información no estructurada tal como texto, audio, video, etc. representa la fuente más grande y de mayor crecimiento en la actualidad, tanto para empresas como para organizaciones gubernamentales. La motivación para el desarrollo de tales frameworks fue construir una plataforma común para el análisis no estructurado, fomentar el reuso de componentes para el análisis y reducir la duplicidad del análisis del desarrollo.

[5] http://uima.apache.org/

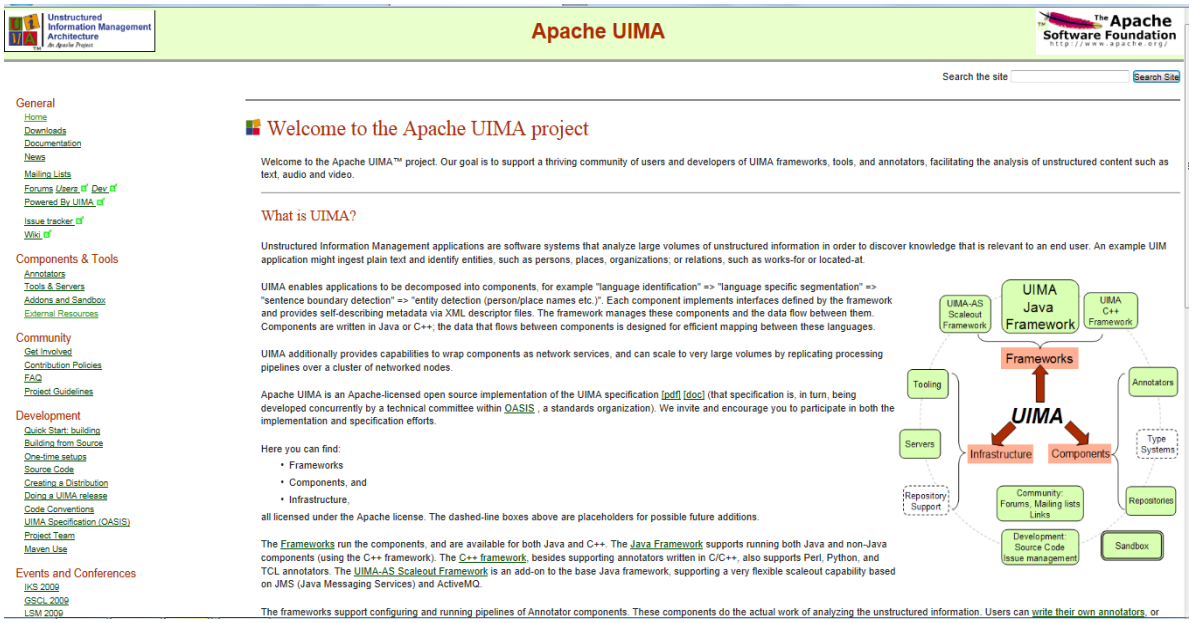

Figura 1. Página principal Apache UIMA.

El propósito de UIMA es permitir que desarrolladores de todo el mundo y diferentes comunidades, desarrollen componentes analíticos complejos, interaccionen cuando sea necesario, compartan, combinen y ejecuten en conjunto dentro del framework proporcionado.
Dentro de la página principal de UIMA se presentan diversas operaciones para los usuarios las mismas están organizadas en: General, Components & Tools, Community, Development, Events and Conferences y ASF. Las de mayor utilidad para los usuarios son tal ves, Development y Components & Tools, (Figura 2).

Figura 2. Opciones disponibles para desarrolladores UIMA.

La primer opción provee información necesaria para iniciarse en el desarrollo reusando componentes UIMA, descargas e instalación, convenciones sobre el código fuente en Java y C++, especificaciones UIMA y guía para el uso de los repositorios.
En la segunda opción es posible acceder a diferentes componentes para el análisis de información no estructurada (Figura 2), herramientas y servidores, Addons and Sandbox son espacios de trabajo a quienes quieran aportar código y deseen unirse a la comunidad UIMA y a recursos externos que están distribuidos en diferentes repositorios UIMA (Figura 3).

Figura 3. Repositorios bajo estándares UIMA

Especificaciones UIMA

Los componentes UIMA deben cumplir con las especificaciones UIMA, que son desarrolladas por un comité técnico dentro de la OASIS (Advancing Open standards for the information society)[6] una organización de estándares. Estos repositorios proveen las herramientas y servidores (Figura 4) necesarios para trabajar en este framework, algunas están incluidas en la aplicación principal, mientras que otras pueden ser descargadas.

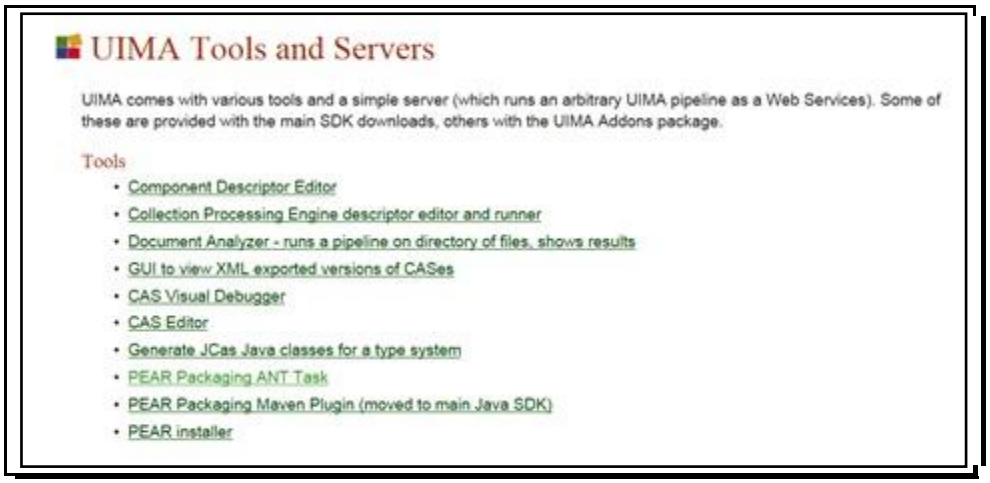

Figura 4. Herramientas disponibles en Apache UIMA.

Entre las herramientas disponibles es posible acceder a un Editor para descripción de Componentes, Editor CAS (Common Analysis Structure) una estructura de datos que contiene información sobre el componente a analizar, como así también el algoritmo de análisis y el resultado del análisis (anotaciones, arboles, relaciones, entidades, etc.) existen muchas

[6] https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=uima

representaciones posibles de un CAS. Además provee un Generador de clases Java para determinados sistemas.
Para publicar un componente en un Repositorios UIMA es necesario empaquetarlo, para esto se utiliza una herramienta PEAR que forma parte de UIMA SDK. PEAR (Processing Engine Archive) es un estándar de UIMA para empaquetar automáticamente componentes desarrollados, tales como motores de análisis, consumidores CAS, recursos, etc. PEAR emplea el formato estándar de compresión zip y permite reusabilidad, modularidad y desarrollo de aplicaciones basado en componentes. Los componentes desarrollados y empaquetados en entornos locales como PEARs, pueden ser implementados e integrados en diferentes SO.
Algunos de los repositorios UIMA ofrecen un glosario con los conceptos propios del área con la cual están relacionados y marcan la diferencia entre Usuarios del sitio y Desarrolladores, es decir que pueden formar parte de la comunidad desde diferentes lugares.

5.2 SourceCodeOnline

SourceCodeOnline[7] es un sitio web en el cual es posible compartir componentes software reutilizables codificados en diferentes lenguajes de programación, artículos, librerías, etc., se puede acceder a los recursos de diferentes formas, (Figura 5);

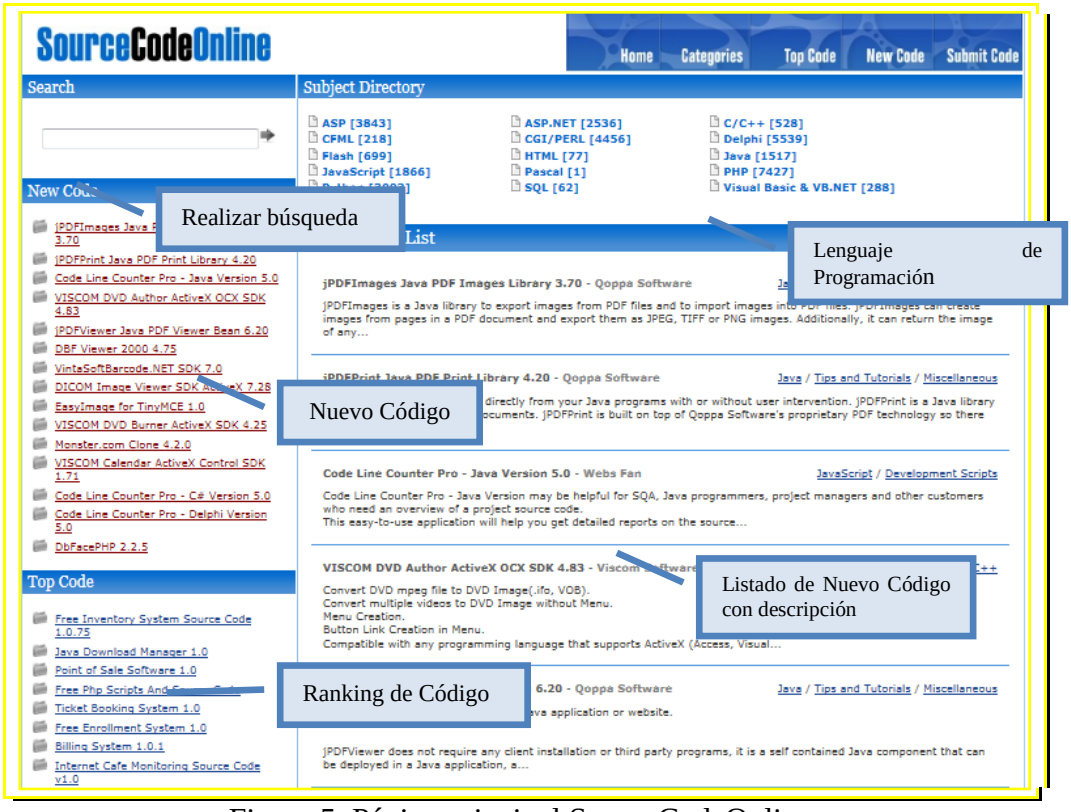

Figura 5. Página principal SourceCodeOnline.

7 http://www.sourcecodeonline.com

Los recursos se encuentran categorizados por lenguaje de programación, ranking de códigos más accedidos, nuevos componentes, etc. Dentro de la categorización por lenguajes de programación se presentan otras subcategorías: Applets, Banner Rotation Scripts, Financial Software, Navigation Scripts, Search Scripts, etc., entre otras, además dentro de cada una de ellas pueden existir nuevas, creando una jerarquía de componentes.
El usuario puede realizar la búsqueda ingresando palabras clave, como resultado de estas búsquedas, el sitio arroja un listado de todas las coincidencias, estos componentes pueden estar disponibles en este repositorio o en otros, en tales casos incluye el link para el acceso. Además provee una breve descripción, nombre del autor, link y categoría a la cual pertenece el componente dentro de las definidas en el sitio (Figura 6). Por otro lado se visualizan resultados de búsquedas relacionadas o similares.

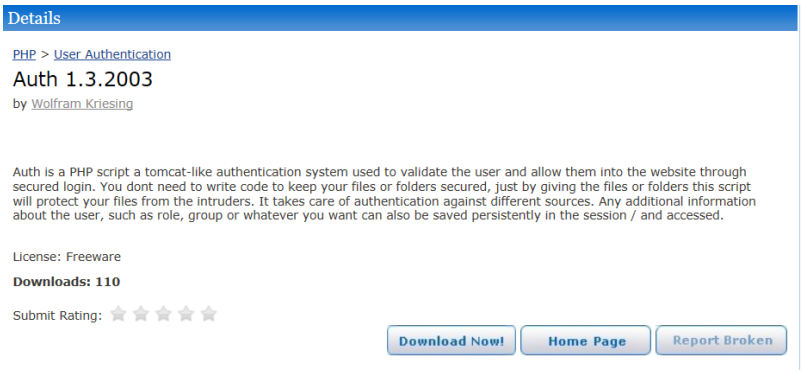

Figura 6. Descripción de un componente.

Para el intercambio de componentes en este sitio no es necesario registrarse, se encuentra disponible la operación Submit Code, la misma permite acceder a un formulario (Figura 7), en el cual se debe volcar la información que describe el componente a enviar, algunos de estos datos deben seleccionarse de listas desplegables, que permiten seleccionar lenguaje y categoría. Luego se debe ingresarse el nombre del componente, versión, una breve descripción, una descripción completa, palabras clave, Sistema Operativo, Licencia, Precio, tamaño, URL, URL de descarga, nombre del autor y página del autor.

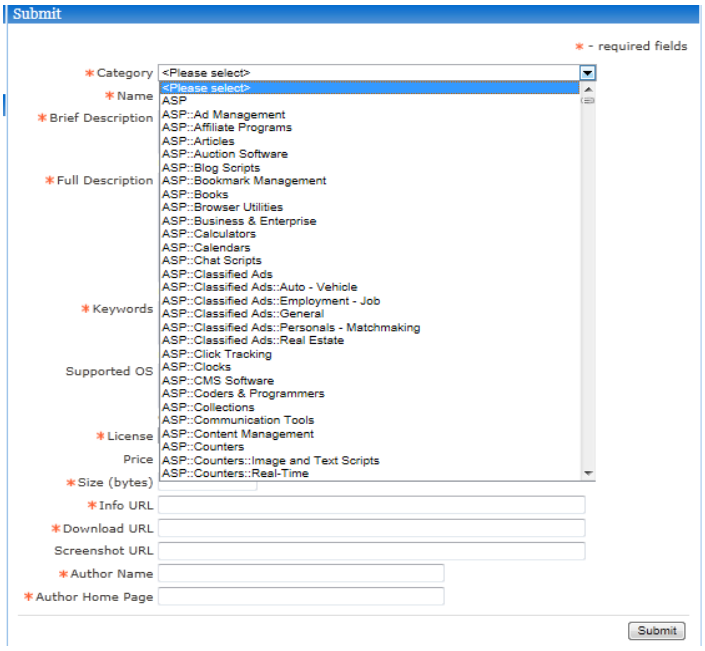

Figura 7. Formulario de publicación de un componente.

5.3 Software artefact Infraestructure Repository (SIR)

El Software artefact Infraestructure Repository (SIR)[8] es un trabajo en progreso, siendo constantemente mejorado y expandido a través de la incorporación de nuevos temas. Luego de registrarse como usuario, es posible acceder a un menú (Figura 8) con opciones tales como, Objetos Software SIR, Publicaciones y Usuarios SIR, Objetos archivados (versiones anteriores), términos de licencia sobre los artefactos.

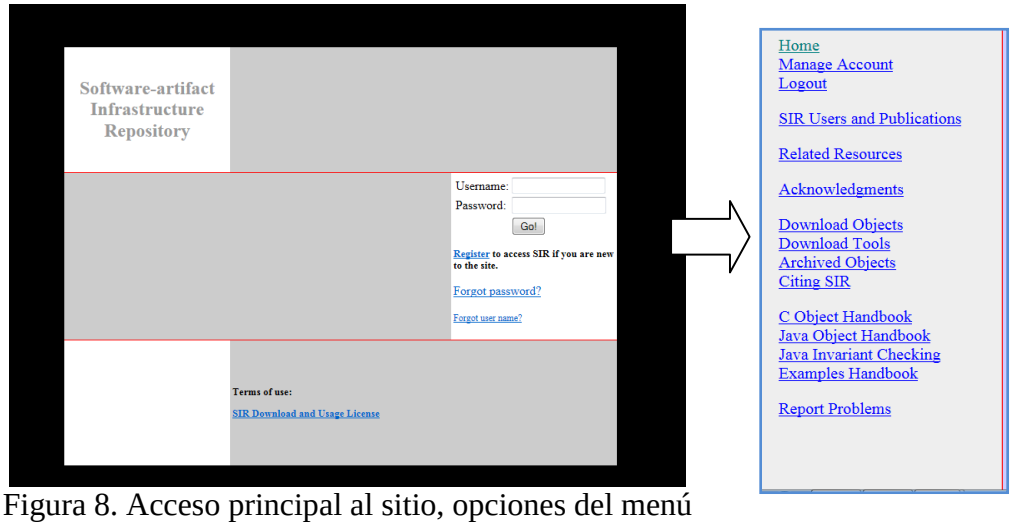

Figura 8. Acceso principal al sitio, opciones del menú principal.

8 http://sir.unl.edu/php/index.php

Es utilizado por Universidades en sus proyectos de investigación y citado en numerosas publicaciones (Figura 9).

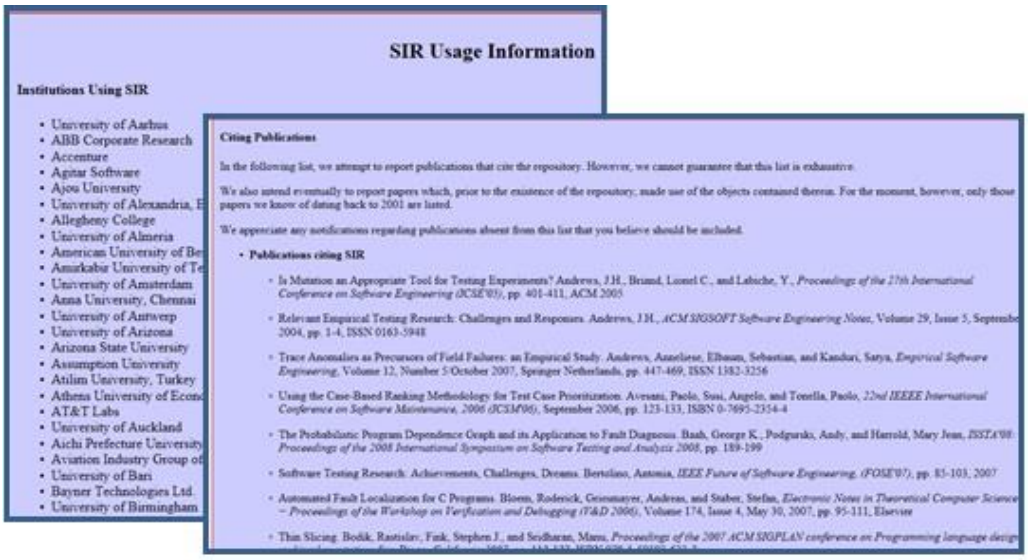

Figura 9. Universidades e Instituciones usuarios – Publicaciones en las cuales es citado SIR

Los componentes de este repositorio cuentan con un Handbooks en el cual se describe como fueron creados y proporciona información sobre su uso (Figura 10).

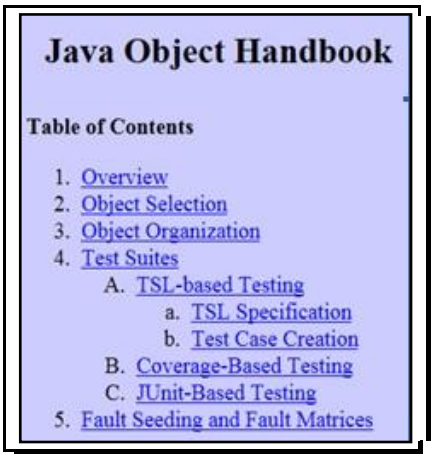

Figura 10. Contenidos de Handbook.

Al seleccionar la opción Objetos Software del menú principal, se accede un listado con todos los componentes del repositorio, estos están numerados, luego ordenados por nombre, se indica el lenguaje, tipo de prueba, tipo de falla, concurrency subject, versión, tamaño en líneas de código, cantidad de procedimientos, última actualización, cantidad de descargas, número de versión dentro de SIR (Figura 11).

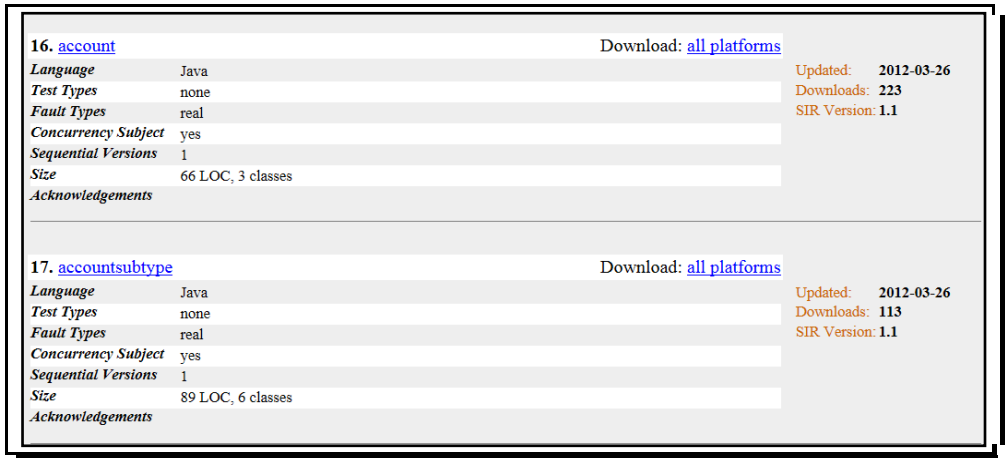

Figura 11. Descripción de componentes en SIR.

En la parte superior permite realizar la búsqueda del componente requerido, seleccionando el lenguaje y nombre del componente (Figura 12), a esta consulta puede agregarse información como tipo de prueba y tipo de falla para una mayor precisión en la búsqueda. En la opción búsqueda avanzada (Figura 13), se habilitan campos como por ejemplo, valores máximos y mínimos de versión, tamaño, clase, matriz de fallas, etc.

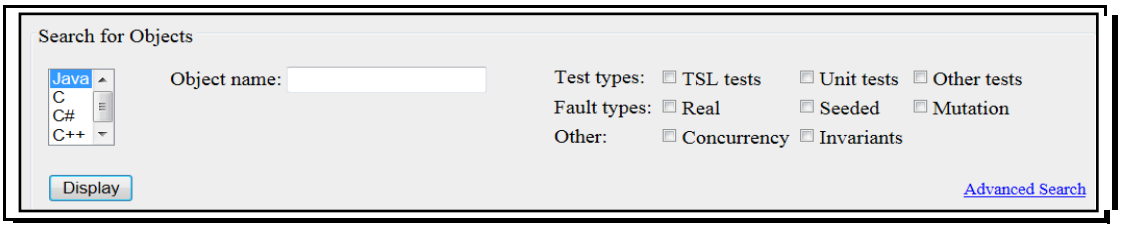

Figura 12. Búsqueda simple.

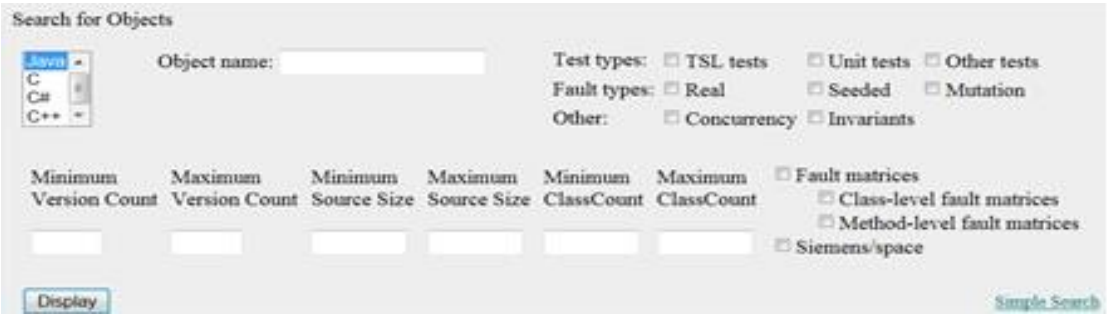

Figura 13. Búsqueda avanzada.

Este repositorio no permite publicar componentes de manera directa, la interacción se realiza a través de mail o por medio de la opción Report Problems.

5.4 VCLComponents.com

VCLComponents.com[9] es un sitio en el cual es posible acceder a gran variedad de componentes, scripts y código fuente implementado en diferentes lenguajes de programación. Estos activos pueden ser free o shareware, cuenta con una barra de menú en su página principal en la cual ofrece diferentes maneras de acceder a los componentes. Por otro lado, los componentes están organizados en un directorio de lenguajes de programación como por ejemplo, ASP, Assembler, Basic, C#, C/C++, entre otros, (Figura 14).

Figura 14. Página principal de VCLComponents.com

Así mismo, dentro de cada lenguaje es posible identificar otras categorías, por ejemplo, para el lenguaje Java, existen las subcategorías Applets, Calendar Scripts, Clock Scripts, Game software, etc.

Algunas de las opciones del menú se destacan en la página principal del repositorio, como por ejemplo: últimos componentes publicados, mejores clasificados, nuevos componentes, componentes más populares, etc., también es posible obtenerlos por nombre de su autor, no es necesario estar registrado para recuperar o publicar un componente. Una opción disponible dentro de la barra de menú es Search, mediante la cual es posible realizar la búsqueda de componentes ingresando una o varias palabras clave (Figura 15), puede seleccionarse el lenguaje de programación de una lista desplegable.

9 http://www.vclcomponents.com/

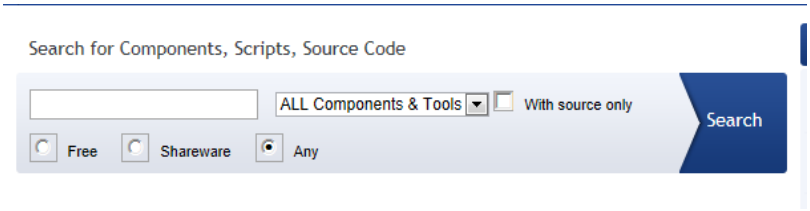

Figura 15. Búsqueda simple.

Los resultados obtenidos de esta búsqueda son listados en páginas, en las cuales se especifica el nombre del componente, licencia, tamaño, una breve descripción del mismo y en la parte inferior se indica lenguaje de programación y categoría a la cual pertenece.
Al seleccionar un componente se accede a mas información relacionada al mismo, como versión, cantidad de descargas, visitas, precio, se indica si incluye o no el código fuente (Figura 16).

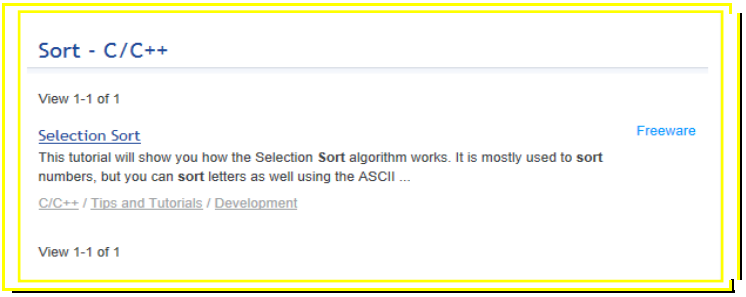

Figura 16. Descripción de un componente.

Para enviar o publicar un componente es necesario completar un formulario (Figura 17) el cual puede dividirse en tres partes, la primera describe al componente, requiere indicar el directorio en el cual se almacenará el componente, el título o nombre del componente, una breve descripción, una descripción más amplia y un grupo de palabras clave.

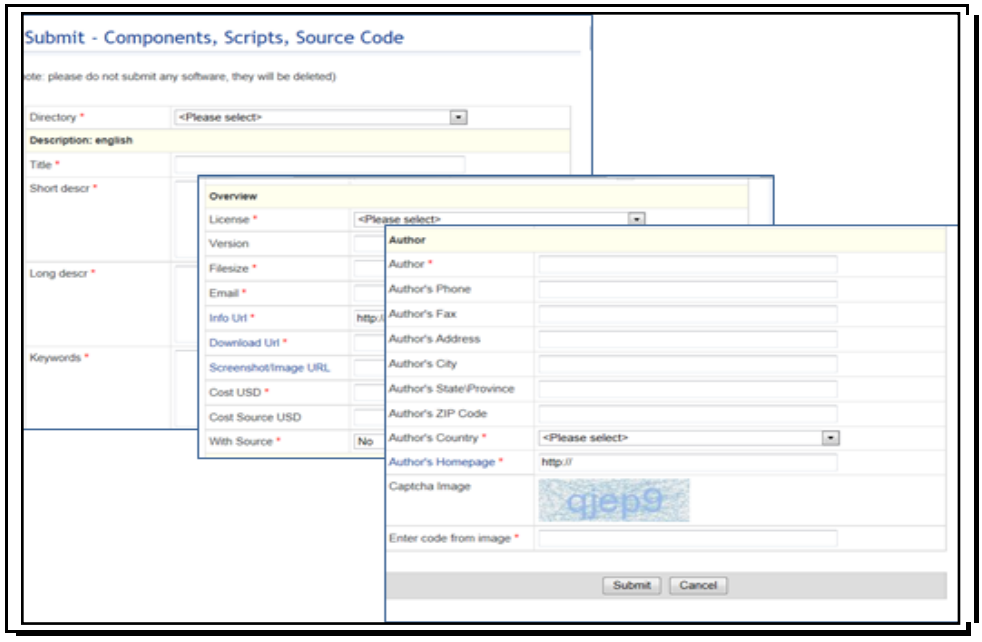

Figura 17. Formulario de publicación de un componente.

La segunda aporta datos del componente tales como licencia, versión, tamaño, e-mail, Url, Url de descarga, precio, además se debe indicar si incluye o no el código fuente. Y la ultima datos del autor: nombre, teléfono, fax, domicilio, ciudad, página web, etc.

5.5 NetLib

NetLib[10] es un repositorio de software libre, papers y databases para matemática, computación científica y otras comunidades. Esta bajo el mantenimiento del laboratorio AT&T Bell, perteneciente a la Universidad de Tennessee. Esta colección de componentes esta replicada en varios sitios de todo el mundo, los cuales están sincronizados automáticamente para proveer un servicio eficiente y fiable a través de la red.
En su página principal,(Figura 18) se presentan los recursos, servicios e información relacionada al repositorio. El acceso puede ser a través de la opción Browser o Search. En cuanto a los servicios que proporciona se pueden mencionar NA net Digest archives, una colección de artículos sobre temas relacionados al análisis numérico y sus prácticas y LAPACK (Linear Algebra Package) una colección de presentaciones, software, documentación, guías de usuarios, etc.

Figura 18. Página principal de Netlib Repository.

10 http://www.netlib.org/

En cuanto a los recursos disponibles, la primera opción browse muestra una lista ordenada alfabéticamente de términos relacionados a los componentes disponibles (Figura 19), ya sean palabras clave o el lenguaje de programación en el cual están implementados.

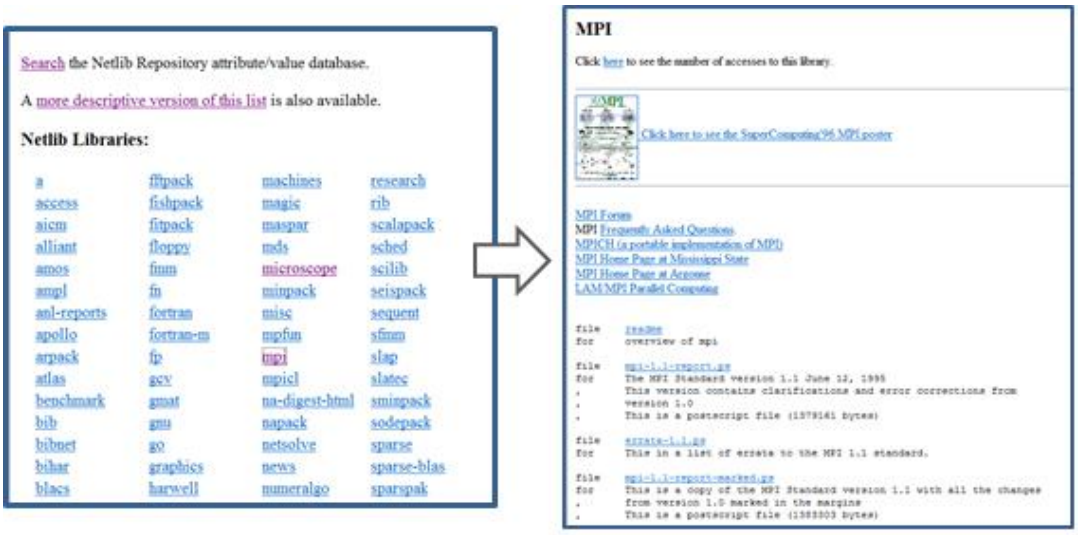

Figura 19. Listado alfabético de recursos disponibles.

En esta página se pueden realizar consultas seleccionando atributos de los componentes como criterio de búsqueda, otra forma es seleccionar uno de estos términos, lo que permite acceder a un listado de archivos con una breve descripción, el nombre del autor, tamaño, etc. Cada componente está definido a través de diferentes atributos, estos son denominados campos globales.

La segunda opción de la página principal, search permite realizar consultas simples o avanzadas, para el primer caso, el usuario ingresa una o varias palabras y luego selecciona si la búsqueda debe incluir todas o al menos una de las palabras (Figura 20. a). Los resultados son presentados en páginas, en cada una son numerados, luego se especifica el nombre, una descripción del objetivo del componente, autor, tamaño y puntuación.

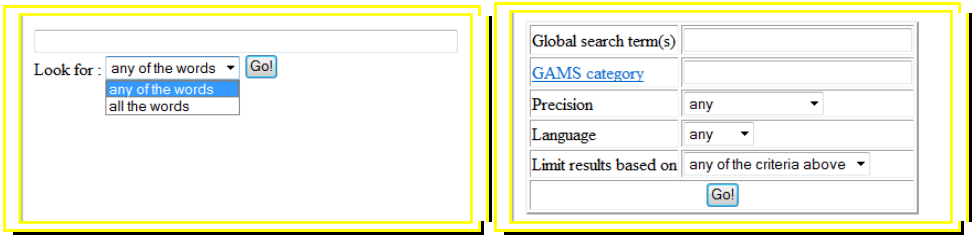

Figura 20 a) búsqueda simple b) búsqueda avanzada

En la parte superior es posible seleccionar otras interfaces de búsqueda, la opción avanzada presenta un formulario (Figura 20 b) con los siguientes campos a ingresar: términos globales de búsqueda, categoría GAMS (Guide to Available Mathematical Software) (Figura 21), precisión, lenguaje de implementación y se define si la búsqueda debe considerar todos los criterios seleccionados o al menos alguno.

Figura 21. Categorias GAMS.

Este repositorio proporciona ayuda en línea con definiciones y ejemplos para llevar a cabo este tipo de consultas sobre la base de datos, utilizando la sintaxis correcta (Figura 22).

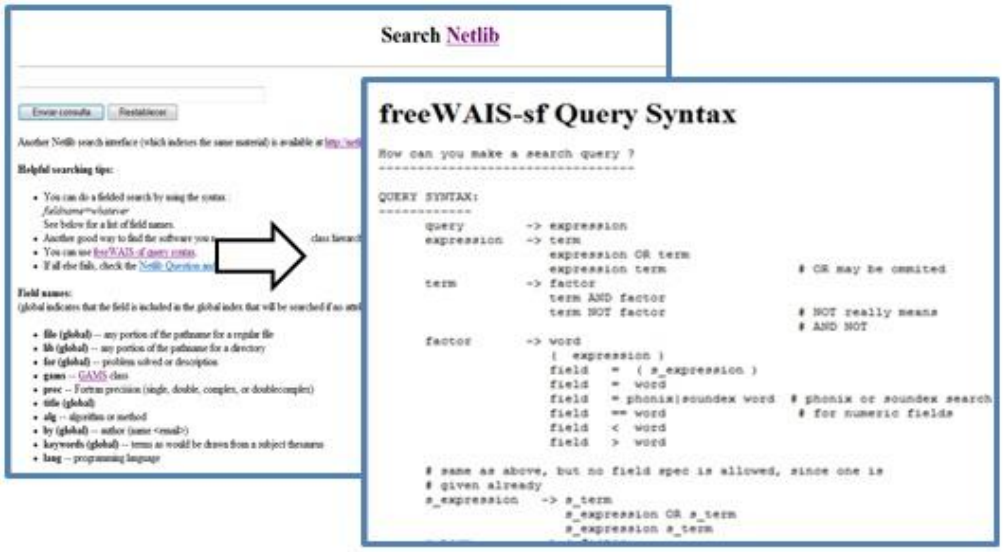

Figura 22. Ayuda en línea para realizar consultas.

Dentro de la ayuda está disponible el acceso de freeWAIS-sf Query Sintax (Figura 22), donde el usuario puede obtener la sintaxis de consultas y ejemplos, también existe un foro de discusión Netlib.

5.6 Otros Repositorios

Existe gran variedad de repositorios, algunos muy populares y muy utilizados desde hace un tiempo, pero no fue posible analizarlos debido a que actualmente no se encuentran disponibles, algunos de ellos se describen a continuación:

7.2 Componex(Component Nexus)[43]: Un modelo para componentes software comerciales, que hizo frente a dificultades y limitaciones en el desarrollo de aplicaciones, facilitando el intercambio entre vendedores y compradores de componentes.

7.3 Repositorio Universal (The Universal Repository)[44]: desarrollado por Unisys[11], basado en principios orientados a objetos, seguía el Modelo de Servicios de Repositorios (RSM) permitiendo recuperar, herramientas, sistemas para la gestión de bases de datos, lenguajes de programación, etc. Los usuarios podían extender el Repositorio Universal incorporando sus propios modelos.

7.4 Component Capitol™[45]: es un repositorio desarrollado por Objects Components Corporation (OCC) para almacenar componentes implementados en lenguajes orientados a objetos tales como Java[47], C++[48], Eiffel[49] y SmallTalk[50].

6. ANALISIS PARA UN REPOSITORIO DE ASPECTOS

Sobre cada uno de los repositorios expuestos en Sección 4, se han analizado los siguientes criterios:

a) Publicación/recuperación de aspectos; si se observa en sus contenidos la existencia de estos módulos.

b) Lenguajes de Programación Orientados a Aspectos (POA)[51]; si incluye dentro de sus categorías lenguajes tales como AspectJ[52], JasCo[53], AspectC++[54], etc.

c) Formulario de publicación; si es necesario cumplimentar un formulario para publicar un componente.

d) Plantilla de especificación de componente; si se utiliza una plantilla o archivo específico para describir el componente a publicar.

e) Tipos de búsqueda que ofrece; simple, avanzada u otras.

f) Restricciones sobre los lenguajes de programación; si es amplio o admite uno específico.

g) Criterios de presentación; categorías, autor, orden, fechas, etc.

h) Método de publicación y recuperación; si es posible identificar alguno de los mencionados en la Sección 3.

i) Funcionalidad de aspectos; si dentro de las categorías definidas en cada uno de los repositorios, pueden identificarse requerimientos no funcionales que puedan ser encapsulados en un aspecto, tales como logging, persistencia, gestión de memoria, seguridad, etc.

UIMA Apache

En el sitio Apache UIMA no se especifican categorías ni lenguajes de programación, no se trata de un repositorio, sino que provee el acceso a diferentes repositorios que comparten especificaciones, conversiones de código, herramientas, etc. Los componentes disponibles son utilizados para el análisis y procesamiento de información en diferentes formatos y dominios.

[11] www.unisys.com

Apache UIMA cumple con estándares definidos por OASIS, lo que define el formato con el cual deben ser publicados los componentes, como así también la especificación en archivos XML. Permite realizar búsquedas utilizando palabras clave. La no restricción sobre los lenguajes de implementación de los componentes, es de gran ventaja ya que no limita la inclusión de los aspectos. En los repositorios explorados no fue posible identificar categorías relacionadas a crosscutting concerns. De acuerdo los requerimientos y estándares definidos para su publicación es probable que se utilicen métodos basados en taxonomía de componentes.

SourceCodeOnline

En SourceCodeOnLine no se identificaron crosscutting concerns ni lenguajes de programación orientados a aspectos dentro de sus categorías, permite realizar búsquedas simples, basadas en una o varias palabras. En este repositorio es necesario completar un formulario con información relacionada al componente y a su autor, pero no requiere plantilla adicional para describir el componente. En cuanto a los lenguajes de programación, se observaron dieciséis lenguajes diferentes, por lo que es bastante amplio y podría extenderse. Por otro lado ofrece diferentes criterios para mostrar los componentes al usuario, dentro de las categorías disponibles se puede acceder a Loggin Accesses and Statistic, Debugging and Error Handling, Replication y Monitoring, que pueden estar relacionadas a requerimientos no funcionales. De acuerdo a la forma de publicar y recuperar los componentes, se considera que están siendo aplicados métodos basados en texto simple.

SIR

Este repositorio provee únicamente programas implementados en C y Java, por lo que es bastante específico, lo que indica que no cuenta con lenguajes orientados a aspectos. Los componentes se categorizan únicamente en Objects, Tools y Archived Objects. Permite filtrar los componentes por lenguaje de programación y realizar búsqueda simple y avanzada. Esta última incorpora varios campos de datos específicos sobre el componente, por lo que su descripción estaría especificada bajo un formato particular. Este sitio no ofrece una opción en la cual el usuario pueda publicar un componente, por lo tanto no se existen formulario ni plantilla a tal fin. Dentro de sus componentes se identifican módulos diseñados para la Detección de errores y Concurrencia los que pueden ser categorizados como crosscutting concerns. Teniendo en cuenta los datos requeridos en la búsqueda avanzada, se cree factible la utilización de métodos basados en la taxonomía de los componentes.

VCLcomponent

En este sitio es posible acceder a componentes implementados en once lenguajes de programación, ninguno de ellos es orientado a aspectos. Permite realizar búsquedas simples, con una o varias palabras además de la categoría. Los recursos están presentados a los usuarios utilizando diferentes criterios, Categorías, Lenguaje de Programación, Autor, Nuevos Componentes, etc. Dentro de sus categorías se distinguen Replication, Monitoring y Security, las cuales pueden incluir requerimientos no funcionales, sin embargo algunos de estos directorios se encuentran aún vacíos. Para enviar un componente es necesario cumplimentar un formulario con la descripción del componente en lenguaje natural, por lo que se estarían aplicando métodos basados en texto simple. No requiere de plantilla para al descripción del componente.

NetLib

NetLib es algo acotado en cuanto a la naturaleza de los recursos que ofrece, pero no presenta restricciones sobre el lenguaje de implementación. Los componentes se pueden acceder desde un listado de palabras clave, tema o lenguaje y a través de consultas. Las categorías y subcategorías están definida por letra, número e inciso a,b,c, etc. en algunos casos. Dentro del listado de palabras clave se identifica Performance, la cual permite acceder a información y recursos sobre performance en bases de datos, esta opción estaría relacionada a crosscutting concerns. NetLib, permite realizar búsquedas simples y avanzadas en las cuales se utilizan campos globales que describen al componente, por lo que se consideran aplicables los métodos basados en la taxonomía de los componentes. En este sitio no se observan opciones directas para publicar componentes, la interacción se realiza por mail, por lo que no requiere de formulario o plantilla para la descripción del componente.

Los resultados sobre los criterios analizados se muestran en la Tabla 1, debiéndose destacar que la última columna indica a) aplicable a aspectos; si cada uno de los criterios es adaptable al módulo de un aspecto y la última fila b) admite aspectos; si es posible incluir aspectos en el repositorio.

Tabla 1. Análisis de Repositorios

Repositorio **Criterio**	**UIMA**	**Sourcecodeonline**	**SIR**	**VCLcomponent**	**NetLib**	**Adaptable al módulo aspecto**
Publicación /Recuperación de aspectos	No permite	No permite	No permite	No permite	No permite	--
Lenguajes POA	No presenta	No presenta	No presenta	No presenta	No presenta	--
Formulario de Publicación	No requiere	Si requiere	No requiere	Si requiere	No requiere	Si, incorporando nuevos campos
Planilla de Especificación	Si requiere	No requiere	No requiere	No requiere	No requiere	Si, incorporando información sobre la estructura del aspecto
Tipos de Búsqueda	Simple	Simple	Simple y Avanzada	Simple	Simple y Avanzada	Si, incorporando campos u opciones específicos
Lenguajes	Sin restricciones	Sin restricciones	Únicamente Java C	Sin restricciones	Sin restricciones	Si, agregando los lenguajes específicos
Criterios de Presentación	Indexados, autor	Lenguajes Código Nuevo Top Code, etc.	Lenguaje	Lenguaje, Categoría, Nuevos Componentes, etc.	Índice de términos	Si, incluyendo nuevas categorías y lenguajes
Método de Publicación y Recuperación	Taxonomía de Componentes.	Texto Libre	Taxonomía de Componentes. Búsqueda Incremental	Búsqueda Incremental. Basados en Texto Libre	Taxonomía de Componentes.	Si, con las adaptaciones necesarias de acuerdo al método seleccionado
Requerimientos no funcionales	No se observan	Loggin , Accesses and Stadistic, Debugging and Error Handling, Replication Monitoring	Concurrencia Detección Errores	Replication, Monitoring Security	Performance	Sí, asociando estas categorías al DSOA
Admite Aspectos	Sí, adaptando plantilla de especif. y formulario de publicación.Adicionando búsqueda avanzada.	Sí, incorporando lenguajes POA en sus categorías, incluyendo req. no funcionales, como subcategorías. Agregando campos específicos del aspecto al formulario.	Sí, incorporando lenguajes POA en sus categorías.	Sí, incorporando campos específicos del aspecto al formulario de publicación , incluyendo lenguajes POA en sus categorías.	Sí, incorporando nuevos campos en la búsqueda avanzada.	

En cuanto al contenido de la última fila y columna de Tabla 1 resultan de gran utilidad para el propósito de este trabajo, permitiendo establecer si es posible adaptar, extender o crear un repositorio que almacene aspectos. Según la última fila todos los repositorios observados **son aplicables** a los módulos de aspectos, con ciertas modificaciones sin alterar su actual

funcionamiento. Con respecto a los criterios analizados, en la última columna se observa que, los tipos de búsqueda, lenguajes de programación, funcionalidad de aspectos y criterios de presentación de componentes, **pueden ser aplicados** a los módulos de aspectos sin inconvenientes. Algunas modificaciones sobre la plantilla de especificación y formulario de publicación permitirán su inclusión en estos sitios.

7. CONCLUSIONES

En presencia de escenarios muy exigentes en cuanto a los requerimientos de los usuarios para el desarrollo de aplicaciones, se proponen diversas metodologías para dar soporte o abordar los problemas que surgen en este proceso. Un objetivo común de los enfoques DBC y el DSOA es fomentar el reuso de software, lo que permitirá el desarrollo rápido, de calidad y mejora en la productividad de los desarrolladores. Por otro lado el DSOA se enfoca en encapsular aquellos requerimientos no funcionales que se dispersan y entremezclan en el código de las aplicaciones, dificultando su mantenimiento. Por esta razón surge la necesidad de unificar aspectos y componentes en un solo ambiente, con este fin se han explorado repositorios de software en búsqueda de módulos que encapsulen crosscutting concerns. En algunos de los repositorios (SourceCodeOnline, SIR, VCLcomponent y NetLib) se han identificado crosscutting concerns dentro de sus categorías, pero no son presentados a los usuarios como aspectos, es decir no se hace referencia a ellos como parte del DSOA. Ante esta situación se puede afirmar que los crosscutting concerns están disponibles en los repositorios, que pueden ser publicados y recuperados y que por lo tanto son reutilizables. Sin embargo, la posibilidad de que su reutilización sea como módulos aspectos aún no se encuentra disponible. Para que su publicación y recuperación se realice considerando a estos componentes como aspectos reusables, es necesaria su implementación en los lenguajes orientados a aspectos. Además se debe tener presente su modularidad, composición y los conceptos que propone el DSOA de manera que el usuario sea capaz de identificarlos y recuperarlos, teniendo conocimiento de que está reutilizando un aspecto.

La publicación y recuperación de componentes es tal vez, el punto clave para determinar si es posible un repositorio que almacene aspectos. Si bien estos son módulos independientes, no deberían ser tratados como simples componentes, debido a que poseen una estructura específica que permitiría realizar una reutilización total o parcial (reuso de pointcuts y/o advices). El método de publicación y recuperación debe tener en cuenta estas posibilidades, considerando al código fuente del aspecto, como el componente a ser reutilizado, por ello requiere adaptaciones sobre la plantilla de descripción de componentes y/o formulario de publicación de manera que puedan ser admitidos y recuperados en un RC. Estas adaptaciones deben incluir la descripción de pointcuts y advices, o bien especificar campos o palabras clave, tales como clases o nombres de métodos. Además estos nuevos campos deben formar parte de las búsquedas avanzadas para identificar los aspectos a ser recuperados. Incorporar los lenguajes POA más utilizados dentro de las categorías, sería de gran utilidad para acceder a los aspectos, es de conocimiento que estos módulos encapsulan cierta funcionalidad específica, como persistencia, sincronización, gestión de memoria, concurrencia, etc. estas podrían formar parte de una subcategoría dentro de lenguajes como AspectJ, JasCO, AspectC++, etc. Las adaptaciones y/o modificaciones sobre el formulario y la plantilla de especificación, deben ser consideradas por el método de recuperación.

El trabajo a futuro consiste en analizar las herramientas disponibles para la construcción de RC teniendo en cuenta los resultados obtenidos, a efectos de seleccionar aquella que permita

desarrollar la extensión o adaptación para aspectos, basándose en las modificaciones y/o adaptaciones necesarias.

8. REFERENCIAS

5 Sommerville, I. Ingeniería de Software. Prentice-Hall. México, CINVESTAV-IPN: pp 42. (2000).
6 Rafael González, Torres. M. Algunas Implicaciones del Desarrollo Basado en Componentes. paper. Bogotá, Colombia: Departamento de Ingeniería de Sistemas Universidad Javeriana; (2005).
7 Jonás A. Montilva C. NAyJAC. Desarrollo de Software Basado en Componentes. paper. Maracaibo , Mérida - Venezuela: IV Congreso de Automatización y Control; (2003)
8 Gregor Kiczales, John Lamping, Anurag Mendhekar, et al. Aspect-Oriented Programming. In: Springer-Verlag, editor. Finland.: European Conference on Object-Oriented Programming; pp25.(1997)
9 Brichau JaT, D'Hondt. Aspect-Oriented Software Development. Paper. (2005.
10 Johan Brichau (VUB), Theo D'Hondt (VUB), Awais Rashid (ULANC), et al. Curriculum Fundamentals of AOSD . Integración: AOSD – EUROPE.(2006).
11 Filman, R., et al., *Aspect-Oriented Software Development*. 2004: Addison-Wesley.
12 Szyperski, C., Component Software: Beyond Object-Oriented Programming, Addison-Wesley (1998).
13 Kitchenham, B. Guidelines for performing Systematic Literature Reviews in Software Engineering Version 2.3. EBSE Technical Report EBSE-2007-1. Software Engineering Group School of Computer Science and Mathematics Keele University. Keele University, Keele Staffs. ST5 5BG, UK y Departament of Computer Science University of Dourham. (2007)
14 Mili, A., Yacoub, S., Addy, E. & Hafedh, M., Reuse Based Software Engineering, John Wiley & Sons, INC.(2002).
15 Nedhal A., Al Saiyd, Intisar A. Al Said, Takrori AHA. Semantic-Based Retrieving Model of Reuse Software Component: Computer Science Department (2010).
16 Heineman, G., Councill, W, Component-Based Software Engineering, Addison Wesley, (2001).
17 Pressman R., Software Engineering A Practitional's Approach, Sixth Ed., McGraw Hill, (2005).
18 Kuljit Kaur PK, Jaspreet Bedi, and Hardeep Singh. Towards a Suitable and Systematic Approach for Component Based Software Development. World Academy of Science, Engineering and Technology (2007).
19 Shailendra Sh., Rajesh B. Storage and Retrieval using Aspect Oriented Tecnique. Proceedings of National Conference on Challenges & Opportunities in Information Technology (COIT-2007) RIMT-IET, Mandi Gobindgarh. (2007)
20 DePrince, W. y Hofmeister, C., Usage Policies for Components en Crnkovik, I., et al (Eds.) Proceedings of the 6th ICSE Workshop on Component-Based Software Engineering: Automated Reasoning and Prediction, Portland, Oregon, USA, May 3-4, 2003, Carnegie Mellon University (2003).
21 Blais, J. J. a. C. Software Hardware Asset Reuse Enterprise. Monterrey , California, NAVAL POSTGRADUATE SCHOOL: pp.75.(2009)

22 Mili, A., R. Mili, and R.T. Mittermeir, A survey of software reuse libraries. Annals of Software Engineering, 5(0): p. 349 – 414.(1998)
23 Fernández-Chamizo, C., et al. Supporting Object Reuse through Case-Based Reasoning. European Workshop on Case-Based Reasoning. Lausanne, Switzerland. (1996).
24 Iribarne, L., J.M. Troya, and A. Vallecillo. Selecting software components with multiple interfaces. 28th Euromicro Conference. (2002)
25 Klein, M. and A. Bernstein. Searching for Services on the Semantic Web Using Process Ontologies. The First Semantic Web Working Symposium. Stanford, CA, USA. (2001)
26 Frakes, W.B. and B.A. Nejmeh, Software reuse through information retrieval. ACM SIGIR Forum, 21(1-2): p. 30-36. (1987)
27 Helm, R. and Y.S. Maarek. Integrating information retrieval and domain specific approaches for browsing and retrieval in object-oriented class libraries. Conference on Object Oriented Programming Systems Languages and Applications. Phoenix, Arizona, United States. (1991)
28 Magnini, B., Use of a lexical knowledge base for information access systems. Journal of Theoretical & Applied Issues in Specialized Communication, 5(2): p. 203 - 228. (1999)
29 Lindig C. Concept-Based Component Retrieval. paper. Braunschweig: Abteilung Softwaretechnologie Technische UniversitÄat Braunschweig; (2007).
30 Aamodt, A. Explanation-driven case-based reasoning. Topics in Case-based reasoning: Springer Verlag. (1994)
31 Mingyang Gu, A. A. a. X. T. Component Retreival Using Conversational Case-Based Reasoning, Department of Computer and Information Science, Norwegian University of Science and Technology**:** pp.12.(2009)
32 Aha, D.W. and L.A. Breslow, Conversational Case-Based Reasoning. Applied Intelligence, 14(1): p. 9-32.(2001)
33 Quanlan, J.R., Induction of decision trees. Machine Learning, 1(1): p. 81-106.(1986)
34 Maninder, S.-S., Goel. Identifying Asset Type for Various Reusable Component Storage/Retrieval methods. Mandi Gobindgarh, Proceedings of National Conference on Challenges & Opportunities in Information Technology - RIMT-IET**:** pp.38-41.(2007)
35 Oualid Khayati, Jean-Pierre. G. Components retrieval systems. France: Laboratoire LSR – IMAG BP 72; 2002 Agosto (2002).
36 Carma McClure. Software Reuse. A Standards- Based Guide. Software Engineering Standards Series. IEEE Computer Society. ISBN 0-7695-0874-X.
37 Rungratchakanon , Usa & Haddad Hisham. Study of information retrieval Systems and Software Reuse Libraries . Kennesaw State University, GA 30144.
38 G. Forbes, McCartan C., Ruairi O'Donnell, Sweeney Niall and Ruben Leon. The integration of information retrieval techniques within a software reuse environment . Department of Information Science, Strathclyde University, 26 Richmond St., Glasgow, G1 1XH, UK.
39 Girardi, M.R. Classification and Retrieval of Software through Their Description in Natural Language, Technical Report 2782, University of Geneva, Geneva, Switzerland (1995)
40 Pahl, C. Ontology-based description and Reasoning for Component.-based Development on the Web en Proceedings of Specification and Verification of Component Based Systems (SAVCBS'03) –ACM SIGSOFT Symposium on the Foundations of Software Engineering Workshop, Sept. 1-2, 2003, Helsinki,Finland, pp. 84- 87, ACM, (2003).

41 Hans-Jörg Happel and Stefan Seedorf, Applications of Ontologies in Software Engineering Lindeberg, T., feature detection with automatic scale selection. International Journal of Computer Vision, 30: 77-116, (2004).
42 Dragan Gaˇsevi´, Nima Kaviani, and Milan Milanovi, Ontologies and Software Engineering (2008).
43 Collaborative Software engineering chapter 6: Application of Ontology in Collaborative Software Development, Springer-Verlag, (2010).
44 Su W., Wang J. and Lochovskyode F. H.: Ontology- Assisted Data Extraction.
45 Klein K., Defining Software Component Specifications: An Open Approach, NDIA Systems Engineering Conference October 22-26, www.dtic.mil/ndia/2007systems/Wednesday/AM/.../5526_Kelin.pdf (2007)
46 F. Giunchiglia, P.Shvaiko, et al. S-Match: an algorithm and an implementation of semantic matching. In Proceedings of ESWS'04. (2004)
47 Overhage S and Thomas P. CompoNex: A Marketplace for Trading Software Components in Immature Markets. K. Czarnecki et al(Eds): pp.145-163 Transit Erfurt. NODE (2003)
48 http://www.uni-ulm.de/~sbauer/programming/OOinfo/FAQ/oo-faq-S-8.13.0.13.html (2012)
49 http://www.manta.com/c/mm7x97n/object-components-corporation (2012)

50 Grundy J. Storage and retrieval of Software Components using Aspects. Department of Computer Science, University of Auckland Private Bag 92019, Auckland, New Zealand. (2000)
51 Sitio web de Java www.java.com.es (2012)
52 http://es.wikipedia.org/wiki/C%2B%2B (2012)
53 http://es.wikipedia.org/wiki/Eiffel_(lenguaje_de_programaci%C3%B3n) (2012)
54 http://es.wikipedia.org/wiki/Smalltalk. (2012)
55 Gregor Kiczales, John Lamping, et al. Aspect-Oriented Programming. Springer-Verlag. Finland., European Conference on Object-Oriented Programming**:** pp25.(1997)
56 Sitio web de AspectJ www.eclipse.org/aspectj *(2012)*
57 Sitio web de JasCo http://ssel.vub.ac.be/jasco/eclipse_jascodt.html (2012)
58 Sitio web de AspectC www.aspectc.org *(2012)*