Técnicas para la clasificación/recuperación de componentes software reutilizables y su impacto en la Calidad

José Luís Barros Justo
Dept. de Lenguajes y Sistemas Informáticos
de la Universidad de Vigo

Resumen

Este artículo presenta una recopilación de las principales técnicas de clasificación y recuperación de componentes software reutilizables, en uso hoy en día, así como su impacto en el aseguramiento de la calidad de los sistemas informáticos desarrollados mediante su composición.. Los mecanismos de ambos procesos se relacionan directamente al depender del mismo objeto; mientras la clasificación determina una organización especifica de los componentes en la biblioteca, la recuperación debe adaptarse a tal organización. Así pues, las técnicas presentadas deben entenderse como procesos de clasificación y recuperación, y no pueden tratarse por separado. En el artículo se presentan las técnicas más usadas, de las que se ocupa mayoritariamente la literatura. Así mismo, se sugieren las características que deberán investigarse con mayor profundidad en aras de lograr nuevas técnicas que eviten los problemas actuales. Aunque con un lenguaje directo (dado su carácter divulgativo), el artículo presenta los conceptos y técnicas con formalidad rigurosa, y ofrece una extensa lista de referencias sobre la evolución de las diversas técnicas, sus aplicaciones, prototipos, relación con otras técnicas, etc. El objetivo principal de este trabajo es el de ofrecer una visión global del estado del arte en técnicas de clasificación/recuperación, aplicadas a componentes software reutilizables, con énfasis en el aspecto cualitativo de los sistemas

resultantes, ofreciendo una extensa lista de referencias donde el lector interesado podrá encontrar mayor información.

Palabras Clave: clasificación, recuperación, componentes reutilizables, recuperación de información, calidad del software.

1. Introducción

El término *componente software reutilizable* se usa en este artículo en un sentido genérico. Los componentes software reutilizables se refieren no solo a código (procedimientos, funciones, rutinas, módulos), sino también a: sistemas, subsistemas, especificaciones del Análisis, especificaciones del Diseño, test de prueba, documentación para el programador, manuales del usuario, etc. En resumen, un componente software reutilizable puede ser **cualquier producto** del proceso de desarrollo de aplicaciones, incluido el propio proceso (métodos, técnicas, experiencias, etc.).

El proceso de clasificación cataloga un componente software en una biblioteca (base de software, repositorio) mediante una traducción directa del componente a una estructura de representación interna, por la cual el componente podrá ser recuperado en el futuro; a este proceso lo denominaremos "mapeo" (por la palabra inglesa *mapping*). En los mecanismos tradicionales de clasificación para la reutilización de software, la representación interna consiste en un conjunto de términos que describen al componente, de acuerdo a las categorías preestablecidas en el esquema de clasificación o en el lenguaje de representación. Para la asignación de términos (indización) puede utilizarse un lenguaje libre o un vocabulario controlado (específico del dominio). El vocabulario controlado establece el conjunto de términos que pueden utilizarse en el proceso de indización y su significado, algo así como un glosario especializado.

La recuperación de potenciales componentes para su reutilización, consiste en la búsqueda y selección de componentes presentes en el repositorio, que cumplen con las características impuestas por el usuario (restricciones). La búsqueda puede ser hecha mediante la visualización de una jerarquía de componentes en la biblioteca, lo que hemos dado en llamar **hojear** (del inglés *browsing*); o siguiendo un conjunto de enlaces hipertexto (o hipermedia), o por *recuperación lineal.* La recuperación lineal es la actividad típica de los sistemas de recuperación de información, en los que el usuario describe las características del objeto requerido mediante una *consulta*, y el sistema devuelve como resultado una lista de objetos que cumplen total o parcialmente con tales características.

La estrecha relación entre clasificación y recuperación, mediante sus actividades comunes, queda de manifiesto en la Figura 1.

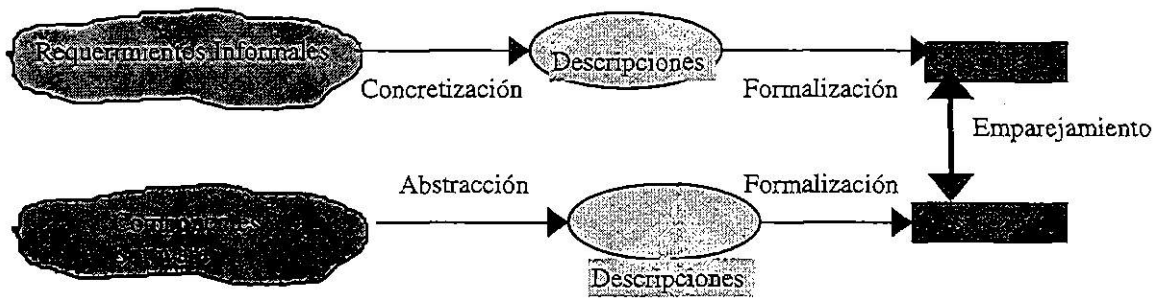


Figura 1 Recuperación y Clasificación

Los mecanismos de recuperación en los sistemas de reutilización también pueden incluir métricas para asegurar el *potencial de reutilización*, y la facilidad de adaptación del componente, e información tal como: frecuencia de reutilización, accesos sin reutilización, accesos y uso sin modificación, uso con adaptaciones, calidad de la documentación, versión, lenguaje, complejidad del componente (tamaño, número de entradas/salidas, etc.).

Es obvio que la reutilización de componentes software implica directamente un incremento en la calidad de los sistemas desarrollados mediante la composición de tales componentes. Por analogía con otras áreas podemos afirmar que el uso de "piezas" (componentes) en repetidas ocasiones (reutilización) ofrece garantías de que la "pieza" carece de errores o "fallos". Cuanto más se reutiliza un componente menor es la probabilidad de encontrar errores en él. Así pués, dentro de las 3 grandes ventajas que ofrece la reutilización de software: aumento de la productividad, incremento en la calidad y disminución del tiempo de entrega, nos quedaremos, en este artículo con la segunda: el impacto sobre la calidad.

2. El Ciclo de Vida con Reutilización

La reutilización de software comprende dos actividades principales[1] -y complementarias-:

1. el desarrollo-para (desarrollar componentes reutilizables)
2. el desarrollo-con (reutilizar componentes existentes)

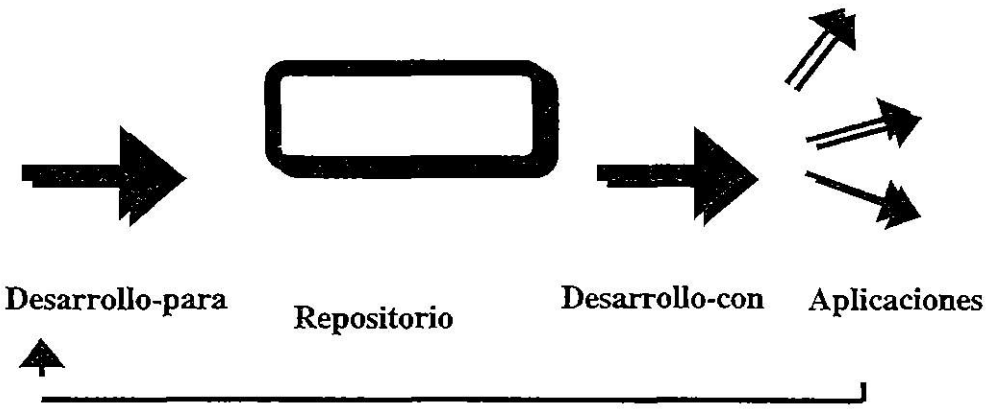


Figura 2 El Ciclo de Vida con Reutilización

El desarrollo-para reutilizar [Morel93] también recibe el nombre de Ingeniería de Aplicaciones [DeMey93], se trata de construir componentes software reutilizables, que puedan ser usados en el desarrollo de aplicaciones similares en un contexto diferente -reutilización horizontal- o aplicaciones diferentes en el mismo contexto -reutilización vertical-.

El desarrollo-con [Casta94], también conocido como Desarrollo de Aplicaciones [Podgu92] comprende las actividades tradicionales de desarrollo de aplicaciones software,

[1] En la literatura también pueden encontrarse las denominaciones de: Acercamiento Generativo [Neigh84] y Acercamiento de Bloques de Construcción [Burto87], para referirse al desarrollo-para y al desarrollo-con, repectivamente.

pero incluye además la búsqueda, selección y adaptación de componentes que se encuentran en un repositorio, para su inclusión en el sistema a desarrollar. Actualmente, el desarrollo-con recibe el nombre de Desarrollo sobre la base de Componentes (Component-based development). Una buena referencia a los procesos de desarrollo-para y desarrollo-con puede encontrarse en [Karls96].

A pesar de los diferentes roles, el desarrollo-para y el desarrollo-con son complementarios (véase

Figura 2). El desarrollo-para es una actividad evolutiva que utiliza la experiencia acumulada durante el desarrollo-con, mientras este último utiliza los componentes desarrollados por el primero. Las actividades en cada uno de los desarrollos pueden verse en la Figura 3 y la Figura 4.

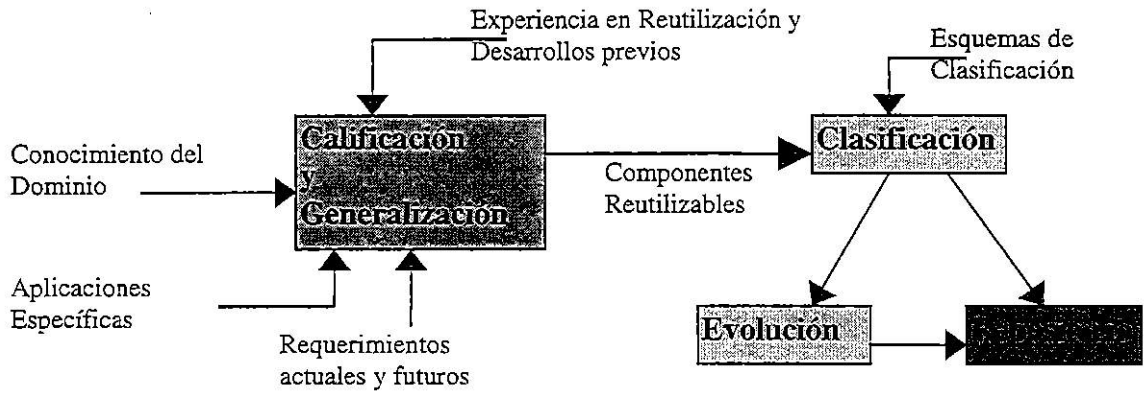


Figura 3 Actividades en el Desarrollo-para

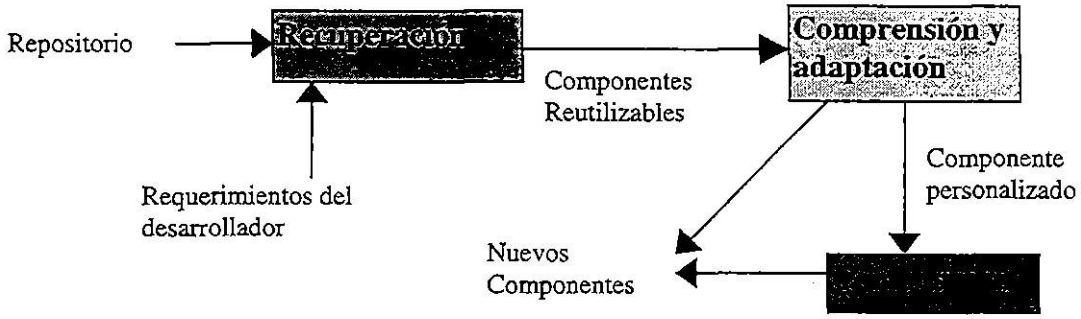


Figura 4 Actividades en el Desarrollo-con

3. Técnicas de Clasificación/Recuperación

Antes de comentar, por separado, cada una de las técnicas actuales de clasificación y recuperación de componentes software reutilizables, indicaremos los conceptos básicos referentes a las diversas actividades. La mayor parte de los conceptos provienen del área de la recuperación de la información [Frake92,Lanca93,Rijsb80,Salto89], y se han adaptado para el caso concreto de la reutilización.

La teoría de recuperación de la información parece ser el acercamiento adecuado a los principales problemas de la clasificación y recuperación de componentes software. A diferencia de lo que ocurre en los sistemas gestores de bases de datos (SGBD, DBMS en inglés), los cuales se basan en la coincidencia total, en la recuperación de información o componentes software es generalmente imposible formular la consulta precisa, y la información recuperada puede incluir ítems que coinciden total o parcialmente con los criterios de la consulta. Los sistemas de recuperación de la información permiten la "coincidencia parcial", lo cual es uno de los requerimientos básicos de cualquier sistema de recuperación, donde la posibilidad de encontrar un componente que coincida exactamente es muy baja.

El propósito de un sistema de recuperación de información (SRI) es el de satisfacer gran variedad de necesidades de información. Una necesidad de información es formulada mediante una consulta al SRI, el cual responderá con una lista (quizás vacía) de ítems en la base de datos. Los ítems -de información- tradicionales son en su mayoría texto (documentos), a pesar de la creciente importancia de la información multimedia: texto, gráficos, sonido, imágenes estáticas y dinámicas, etc. Cuando los ítems en la base de datos son en su mayoría productos del desarrollo de software (requerimientos del análisis, especificaciones del diseño, programas ejecutables, test de prueba, etc.), ésta recibe el nombre de base de software, o repositorio.

La
Figura 5 muestra las principales funciones de un SRI.

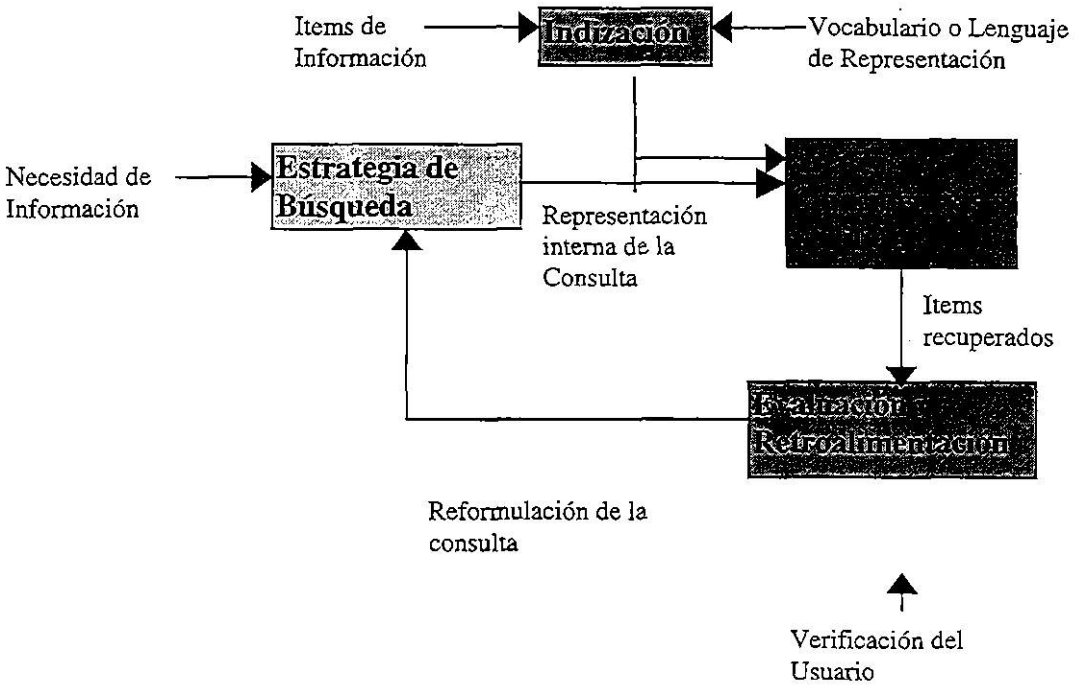


Figura 5 Principales funciones de un SRI

3.1. Indización

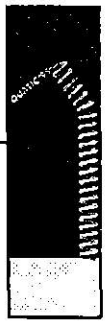
El proceso de indización o catalogación es, esencialmente, un proceso de clasificación realizado mediante un análisis conceptual del ítem de información.

En los SRI sobre la base de palabras clave (los SRI sobre la base de un modelo de espacio vectorial [Salto71]), el proceso involucra la asignación de los ítems de información

a ciertas clases, donde una clase es el conjunto de todos los ítems de información a los cuales se les ha asignado un término particular del índice (palabra clave). Los ítems de información pueden pertenecer a diferentes clases. En algunos SRI los índices (palabras clave) pueden tener asignado un "peso", para reflejar su importancia relativa con respecto al tema central del ítem de información. El conjunto de términos índice asignados a un ítem de información a través del proceso de indización constituye su "representación interna", también conocida como: descriptores, vectores-término, perfiles, etc. En los acercamientos lingüísticos a los SRI, la clasificación consiste en un "mapeo" del ítem de información, descrito mediante un conjunto de oraciones, en una estructura interna de representación del conocimiento (marcos, redes semánticas, etc.).

El proceso de indización puede ayudarse, especialmente en los SRI que lo hacen manualmente, de un vocabulario controlado (específico, restringido), el cual establece el conjunto de términos que pueden ser usados (válidos) para representar tanto el ítem de información como los términos de la consulta. La presencia de un tesaurus, con una clasificación propia del conjunto de términos válidos, es característica de este tipo de SRI. Otros sistemas pueden usar un lenguaje de representación, el cual especifica las características de la representación interna tanto de los ítems de información como de las consultas.

En los SRI tradicionales las consultas se expresan como un conjunto de términos clave (palabras clave), frecuentemente relacionadas con operadores lógicos (Y, O, NO). Los SRI más avanzados permiten formular consultas como frases en un lenguaje natural. La estrategia de búsqueda establece el mecanismo para extraer de la frase -consulta- ciertos términos y transformarlos en una representación interna, similar a la utilizada durante la indización.



Para la indización automática en los SRI se usan, actualmente, dos categorías de métodos:

- métodos estadísticos
- métodos lingüísticos (o, sobre la base del conocimiento)

Los métodos estadísticos [Fagan89,Salto94] se basan en la frecuencia de las palabras en un texto, es decir, las palabras que ocurren con mayor frecuencia en un texto no son útiles en el proceso de clasificación, pues no permiten distinguir -discriminar- los conjuntos de términos de información. Así pues, muchos de los SRI sobre la base de métodos estadísticos poseen una "lista de palabras vacías" que no tienen valor decisivo en el proceso de clasificación y, por tanto, son ignoradas cuando se encuentran en el ítem a clasificar. Además pueden implementarse mecanismos que extraigan de la entrada (en las consultas en lenguaje natural) ciertas construcciones gramaticales, tales como sufijos, prefijos, sinónimos, homónimos, etc. Incluso, puede obtenerse la raíz común de diversos términos y clasificarlos a todos según ella (por ejemplo: *cálculo*, para calcular, calculadora, calculista,...) [Salto83].

Algunas desventajas de los métodos estadísticos pueden resumirse en:

- efectividad reducida en la recuperación (términos fuera de contexto, términos muy generales) [Salto90]
- requieren grandes "muestras" para que el análisis de frecuencia sea apropiado (lo cual no es el caso típico en los componentes software o en los documentos que describen su funcionalidad).

Por otro lado, la investigación más reciente en el campo de los SRI reconoce que algún tipo de tratamiento del lenguaje natural podría mejorar la efectividad en la recuperación de información textual [Lanca93,Salto89,Smeat92]. Es así como surgen los métodos lingüísticos, sobre la base del conocimiento del lenguaje. La idea central es muy clara: la comprensión total de los ítems de texto podría resolver la mayor parte de los

problemas asociados con la recuperación de información. Sin embargo, un acercamiento práctico e independiente del dominio para entender completamente un lenguaje natural no es posible, hoy en día, con las técnicas disponibles de procesamiento del lenguaje natural (PLN), básicamente por el conocimiento necesario y el esfuerzo humano requerido para desarrollar tal aplicación software [Covin94]. Los avances en las técnicas de adquisición de conocimiento, la disponibilidad de diccionarios electrónicos, y las bases de conocimiento contribuirán a extender el uso de los métodos lingüísticos en un futuro cercano [Steel94].

Sin embargo, la información lingüística puede ser extraída de un documento sin necesidad de entender completamente su significado, dado que no es necesario -generalmente- efectuar una interpretación exacta (sin ambigüedad) del texto para asegurar una recuperación "razonablemente efectiva". Por tanto, la integración de ambas técnicas, la estadística y la lingüística, en un solo marco para la recuperación textual ha motivado un creciente interés de los investigadores [Croft93].

3.2. Indización Automática

La indización por texto libre extraer automáticamente palabras clave de las consultas en lenguaje natural y tales términos son usados para localizar los componentes software en el repositorio. Igualmente, los componentes software son clasificados en el repositorio mediante la indización resultante de las palabras clave extraídas a partir de su documentación, previsiblemente en lenguaje natural. La mayor parte de estas técnicas trabajan en el ámbito lexicográfico, por tanto no usan ningún tipo de información sintáctica y/o semántica presente en las descripciones del componente.

Dado que las palabras clave son extraídas de forma automática -de las propias descripciones del componente-, estas técnicas son más "baratas" que las que utilizan

vocabularios controlados, además, pueden hacerse tan específicas como sea preciso y/o aplicarse a cualquier dominio.

Estos sistemas utilizan técnicas estadísticas para recuperar información e indizar, automáticamente, los términos elegidos. Es necesario disponer de gran cantidad de descripciones en forma de texto para que los acercamientos estadísticos funcionen con relativa eficacia. Lamentablemente, las descripciones de componentes software (documentación) no suelen destacar por su tamaño, ni por su claridad, ni por el uso de un lenguaje natural gramaticalmente correcto [Priet91], por ello estas técnicas no resultan muy aplicables a la recuperación de software, aún cuando trabajan relativamente bien en un entorno de información global.

Algunas mejoras han supuesto los siguientes añadidos: a) extracción de pares de palabras clave, de acuerdo a su afinidad lexicográfica y cantidad de información; b) incluir el código fuente, y no tan solo la documentación, buscando relaciones de herencia y parte_de (en software OO), etc. Un caso ejemplar es el de la Biblioteca de Software Reutilizable (RSL por sus siglas en inglés) [Burto87] que incluye el repositorio RSL y cuatro subsistemas: a) el manejador del repositorio, usado para poblar y mantener el repositorio, revisa los ficheros de código y extrae comentarios con etiquetas especiales que se usan como palabras clave para describir la funcionalidad del componente, su autor, la fecha de creación, etc.; b) el sistema de consultas, con una interface de lenguaje natural que interactúa con el usuario para ir refinando la consulta y reducir el conjunto de componentes recuperados; c) el sistema de recuperación y evaluación, que recupera componentes y permite ordenarlos jerárquicamente mediante la especificación de palabras clave que describen los atributos de los componentes; y d) el sistema de diseño asistido por ordenador que permite incluir los componentes recuperados -y seleccionados- en una especificación de diseño OO.

Como conclusión puede afirmarse que los acercamientos de indización automática no tienen una buena relación costo/beneficio para pequeñas colecciones de componentes, sin embargo, comienzan a aparecer en el mercado grandes colecciones de componentes, como

ASSET [Asset], del orden de miles de elementos, que si pueden beneficiarse de estos acercamientos para la generación automática de la representación de los componentes software.

3.3. Facetas

Los principales acercamientos a la clasificación y recuperación proponen un "esquema de clasificación" para catalogar los componentes en una Base de Software (repositorio) [Morel93,Priet91,Sindre93]. Los esquemas especifican algunos atributos, llamados *facetas*, que deben ser usados como descriptores del componente software, generalmente con énfasis en la acción ejecutada por el componente y por los objetos manipulados.

Tanto la clasificación como la recuperación se realiza especificando términos clave para cada atributo en el esquema. Las relaciones entre términos se expresan mediante combinación de atributos, permitiendo una mayor precisión que los sistemas que se basan solo en palabras clave. La aparición de las *facetas* se debe a Prieto-Díaz [Priet91], su esquema fue adoptado o generalizado por muchas otras propuestas en el área de la reutilización [Borst92,Borst93,Oster92]. El esquema consiste de 4 facetas:

- función realizada por el componente (funcionalidad)
- objetos manipulados por la función
- estructura de datos donde la función tiene lugar (medio o localidad)
- sistema al que pertenece la función

Por ejemplo: la familia de comandos *grep* en Unix, podría clasificarse con las siguientes facetas:

- función: buscar
- objetos: cadena de texto
- estructura de datos: fichero
- sistema: editores de líneas

Las tres primeras facetas se consideran suficientes para describir la funcionalidad (comportamiento) del componente. Los catálogos son construidos manualmente y la recuperación se realiza especificando términos (de un vocabulario controlado) para cada una de las facetas. Los componentes recuperados se organizan de acuerdo a un valor de similitud, que es determinado mediante un grafo de distancia conceptual y lógica difusa. Los grafos de distancia conceptual se construyen manualmente para cada dominio de aplicación, sobre la base de la "experiencia, intuición y sentido común" [Priet91].

Un buen ejemplo de clasificación por facetas puede encontrarse en el repositorio de ASSET [Asset], donde se consideran las siguientes: *datos específicos*, *tipo de distribución*, *tipo de componente*, *colección*, *dominio*, *función* y *objeto*. Otras buenas referencias son: Wood y Sommerville [Somme90] con un esquema de clasificación sobre la base de marcos intentando capturar el significado (semántica) de los componentes software, y el proyecto PROTEUS [Frake90], que soporta diferentes métodos de clasificación (palabras clave, facetas, grafos enumerados, atributo-valor) y pretende llevar a cabo una evaluación empírica (comparativa) de tales métodos. Otro interesante esquema de clasificación para repositorios orientados a objeto ha sido propuesto por Girardi y Price [Girar92]. El proyecto REBOOT [Karls96] implementa el esquema de clasificación por facetas que se presenta en la Figura 6.

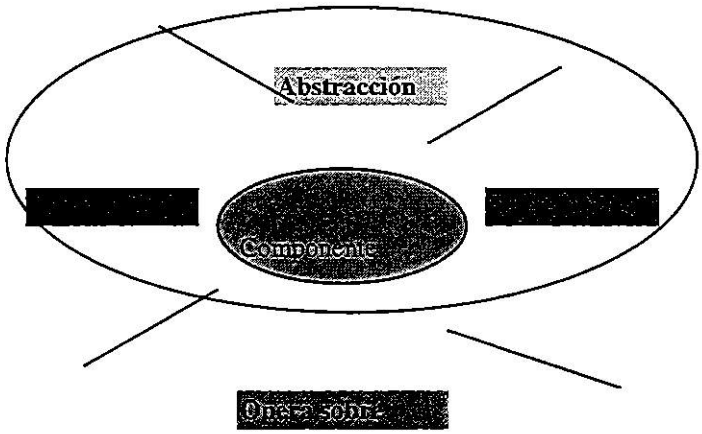


Figura 6 Facetas en el proyecto REBOOT

Donde las facetas representan los siguientes conceptos:

- Dependencias: cuales son las restricciones aplicadas al componente (limitaciones)
- Opera sobre: sobre que objetos actúa el componente (relación, comunicación, etc.)
- Operaciones: que hace el componente (operaciones, funciones, procesos)
- Abstracción: que es el componente (nombre del objeto en un acercamiento OO)

3.4. Marcos de Conocimiento

Estas técnicas se caracterizan por hacer un tipo de análisis lexicográfico, sintáctico y semántico de las especificaciones en lenguaje natural de los componentes software, sin pretender una comprensión completa de los documentos. Se apoyan en una Base de

Conocimientos que almacena la información semántica sobre un dominio de aplicación específico, y sobre el propio lenguaje natural. Estos sistemas son más poderosos que los que usan palabras clave, por contra, requieren enormes recursos humanos para construir las bases de conocimiento -dependientes del dominio- y poblarlas. Algunos de los sistemas más característicos de este acercamiento son los siguientes.

LaSSIE (Large Software System Information Environment) [Devan91], se compone de una base de conocimientos, un algoritmo de análisis semántico sobre la base de inferencia formal, y una interface de usuario que incorpora un visualizador (browser) gráfico y un parser de lenguaje natural (inglés). LaSSIE ayuda a los programadores en la búsqueda de información útil en grandes sistemas software y da soporte a la recuperación de componentes para su reutilización. Aunque la construcción de la base de conocimientos es un proceso manual, actualmente se investiga en la definición de mecanismos de adquisición de conocimiento que automaticen el proceso.

NLH/E (Natural Language Help/English) [Tichy89] es un sistema de pregunta-respuesta. Los autores proponen un catálogo sobre la base de marcos de conocimiento para los componentes software, organizados como una gramática de casos. Los marcos son creados manualmente, pero las consultas son generadas (parser) mediante la gramática de casos. El generador (parser) devuelve el árbol de los marcos emparejados. Si no existe concordancia que consuma toda la entrada, el generador devuelve las "mejores" concordancias, es decir, aquellas que tienen mayor parte de la entrada. El catálogo es específico para el dominio del lenguaje Common Lisp. La entrada (consulta) se restringe a oraciones imperativas o frases nominales.

Nie [Nie93] propone un sistema de recuperación de información donde tanto las consultas como las descripciones de software son generadas (parsed) usando una gramática semántica (para el idioma francés). Se creó una base de conocimientos para un dominio de aplicación específico. El conocimiento consiste en un conjunto de clases que modelan datos y operaciones en el dominio, y un conjunto de relaciones entre los conceptos en las clases.

Allen y Lee [Allen89] proponen un sistema sobre la base de marcos denominado Bauhaus, como soporte a la descripción y recuperación de partes reutilizables y su composición para usar en ambientes de desarrollo con lenguaje Ada. La base de conocimientos contiene descripciones de marcos de partes reutilizables. Métodos manuales y automáticos se utilizan en el proceso de indización.

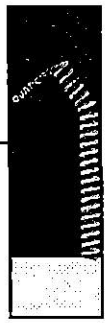
Fitzgerald y Mathis [Fitzg88] construyeron un sistema experto para recuperar componentes en Ada. La base de conocimientos consiste en un conjunto de reglas que relacionan atributos del componente (estructura, ambiente, requisitos de memoria, etc.). El sistema interactúa con el usuario preguntando acerca de estos atributos y, entonces, sugiere un componente o informa al usuario que no se encontró ningún componente que coincidiera.

3.5. Especificaciones Formales

Existen varios sistemas de recuperación de información sobre la base de especificaciones formales [Jeng93,Knigh92,Mitte94,Rolli91,Steig92].

En todo estos acercamientos las consultas son especificaciones formales de requerimientos, y el sistema devuelve los componentes relevantes desde el repositorio, componentes que, a su vez, están especificados formalmente. La recuperación se hace mediante un demostrador de teoremas, para comprobar si las especificaciones del componente satisfacen los requerimientos formales de la consulta.

Entre las ventajas destacadas por los autores figuran: el hecho de que las especificaciones formales se centran en el comportamiento del componente, en lugar de su descripción; que no presentan ambigüedad y que proporcionan mayor precisión que los métodos informales. Por otra parte, otros autores [Steig92] encuentran desventajas, entre las



que cabe citar: a) solo una parte muy pequeña del software disponible para reutilización se encuentra especificado formalmente; b) el tiempo de proceso para los algoritmos de búsqueda puede ser excesivo; c) es más fácil -y barato- usar las descripciones textuales que las especificaciones formales; d) la concordancia parcial, sobre la base de similitud, es muy difícil de controlar, dado que estos acercamientos se basan más en concordancia exacta; y, finalmente, e) el actual estado de la tecnología de demostración-de-teoremas hace imposible la implementación de muchos de estos acercamientos [Podgu92].

Es probable que si el desarrollo formal de software se hace lo suficientemente popular como para justificar la inversión en la construcción de bases de especificaciones formales, estas especificaciones serán la principal fuente de información para las actividades de indización (clasificación). No solo por el hecho de que esta información proviene directamente del componente software (y no de la documentación), sino también porque parece ser la forma más natural de determinar las diferencias entre el componente buscado y el encontrado en el repositorio, y saber así el esfuerzo necesario para adaptarlo (reutilizarlo). El principal problema es que los actuales acercamientos para la recuperación mediante especificaciones formales usan formalismos que tal vez no sean adecuados para lograr una indización efectiva, y por tanto, son insuficientes para permitir una recuperación automática. Tal vez la solución habrá que buscarla por el otro lado del mismo camino, es decir, adaptar los formalismos existentes (o definir nuevos formalismos) para tener en cuenta los problemas de la clasificación y recuperación [Morga91].

3.6. Redes Neuronales

En los sistemas de recuperación sobre la base de un esquema de clasificación, el cálculo de la similitud entre componentes suele hacerse por medio de un grafo de distancia conceptual, el cual establece la similitud entre los términos que describen al componente en su representación interna, pertenecientes -los términos- al vocabulario controlado. El ajuste

que cabe citar: a) solo una parte muy pequeña del software disponible para reutilización se encuentra especificado formalmente; b) el tiempo de proceso para los algoritmos de

búsqueda puede ser excesivo; c) es más fácil -y barato- usar las descripciones textuales que las especificaciones formales; d) la concordancia parcial, sobre la base de similitud, es muy difícil de controlar, dado que estos acercamientos se basan más en concordancia exacta; y, finalmente, e) el actual estado de la tecnología de demostración-de-teoremas hace imposible la implementación de muchos de estos acercamientos [Podgu92].

Es probable que si el desarrollo formal de software se hace lo suficientemente popular como para justificar la inversión en la construcción de bases de especificaciones formales, estas especificaciones serán la principal fuente de información para las actividades de indización (clasificación). No solo por el hecho de que esta información proviene directamente del componente software (y no de la documentación), sino también porque parece ser la forma más natural de determinar las diferencias entre el componente buscado y el encontrado en el repositorio, y saber así el esfuerzo necesario para adaptarlo (reutilizarlo). El principal problema es que los actuales acercamientos para la recuperación mediante especificaciones formales usan formalismos que tal vez no sean adecuados para lograr una indización efectiva, y por tanto, son insuficientes para permitir una recuperación automática. Tal vez la solución habrá que buscarla por el otro lado del mismo camino, es decir, adaptar los formalismos existentes (o definir nuevos formalismos) para tener en cuenta los problemas de la clasificación y recuperación [Morga91].

3.6. Redes Neuronales

En los sistemas de recuperación sobre la base de un esquema de clasificación, el cálculo de la similitud entre componentes suele hacerse por medio de un grafo de distancia conceptual, el cual establece la similitud entre los términos que describen al componente en su representación interna, pertenecientes -los términos- al vocabulario controlado. El ajuste

existe ninguno -al menos conocido por el autor- que permita "marcar" los elementos de interés encontrados durante el proceso y generar un conjunto que pueda ser visualizado posteriormente, algo similar a la agenda (bookmark) de algunos navegadores web. La misma observación se hace para las aplicaciones que implementan la siguiente técnica, hipertexto.

3.8. Hipertexto

Son varias las propuestas para recuperación sobre la base de la tecnología hipertexto [Cybul93]. En tales sistemas la información se organiza como una red de nodos -unidades de información- que se interconectan por medio de enlaces y relaciones. El usuario puede navegar por la red siguiendo los enlaces preestablecidos, y guiándose en este proceso por la semántica de cada enlace [Savoy93].

El principal problema de esta técnica es que el desarrollo y mantenimiento de un repositorio en un ambiente hipertexto requiere de una inversión muy grande en recursos humanos. Otras desventajas apuntan a: a) no existen buenas sugerencias en los enlaces para permitir decidir cual seguir; b) no siempre la red formada por los enlaces es la más adecuada, más aún, es imposible decidir si existe una red óptima; c) el proceso de añadir un nuevo componente a la red es tremendamente complicado, pues deben considerarse todos los posibles enlaces que se relacionen con el nuevo componente; d) en el proceso de eliminar un componente de la red debe asegurarse la destrucción de todos los enlaces que llevaban a él; e) no existe garantía de encontrar lo que se busca, aunque exista un camino que lleve a él, el usuario puede viajar en "círculos" puesto que no es posible explorar todos los caminos (explosión geométrica).

Algunas propuestas muy recientes apuntan al primer problema planteado, tratando de desarrollar ambientes de construcción de hipertextos automáticamente. En estas propuestas los enlaces son generados automáticamente a partir de información adicional que se introduce en la descripción del componente, por ejemplo: el atributo *usa a* <nombre de componente> permite establecer un enlace entre dos componentes (el que *usa* y el *usado*)

[Allan95]. Sin embargo, en nuestra opinión, esta generación automática de enlaces no resuelve el problema, tan solo lo traslada, ahora es necesario invertir el esfuerzo anterior -enlaces- en añadir a las descripciones de los componentes nuevas relaciones, que si no están bien diseñadas ocasionarán la "generación automática de una red inútil". Además, no resuelven el problema que, para el autor, es el inconveniente principal: asegurar que lo que se busca pueda "encontrarse".

4. Impacto en la Calidad

Existen diversos estudios sobre el efecto que han tenido proyectos de reutilización en empresas de todo el mundo, quizás los más tratados en la literatura corresponden a empresas u organismos públicos de la talla de Hewlett-Packard, Motorola, Raytheon, IBM, Microsoft, Nasa, Thomson, Boeing, Siemens, Bull, Petrobras, Dod(USA), etc. En todos ellos se resaltan como efectos positivos los aumentos drásticos en productividad y calidad, así como la reducción de los tiempos de desarrollo. Estos efectos pueden trasladarse directamente a un solo factor: disminución de costos.

En lo que respecta a las técnicas de clasificación y recuperación, su efecto sobre la calidad es indirecto. Por ejemplo, una organización (clasificación) adecuada de los componentes en el repositorio garantizará una recuperación efectiva y eficiente de componentes candidato, proporcionando un valor de precisión alto (componentes muy similares a lo que se busca) mientras se mantiene un valor de recuperación (recall, número de componentes encontrados) adecuado. Por tanto, la calidad asociada a las técnicas de clasificación/recuperación puede entenderse por estos dos criterios:

1. Precisión: calidad de los componentes encontrados, medida como "similitud" entre lo requerido y lo encontrado. Si se encuentra el componente exacto que se buscaba la precisión es del 100%
2. Recuperación (recall): cantidad de componentes encontrados, este valor debe ofrecer un rango que permita al usuario un manejo apropiado del conjunto de componentes recuperado. Pocos componentes recuperados puede ocasionar problemas de

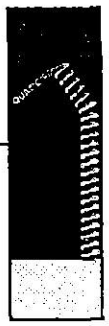
3. adaptación (elección difícil), por el contrario, muchos componentes pueden resultar en un trabajo inabordable por el usuario, quién finalmente termina desechando el conjunto ofrecido, por su intratabilidad.

Existen otros factores, relacionados con la calidad de las propias técnicas y no con los sistemas desarrollados por composición. Entre ellos podemos contar:

1. Calidad de la interface de usuario: en cuanto a poder descriptivo, sencillez de manejo, período corto de curva de aprendizaje, adaptabilidad a las preferencias de los usuarios (colores, menús, barras, etc.).
2. Calidad del esquema de clasificación: en cuanto a la correcta descripción de los componenets y la asignación de un único lugar en la organización, lograr que los componentes similares se organicen en zonas próximas del esquema, no ambigüedad en los criterios de clasificación (objetividad), etc.
3. Calidad del proceso de recuperación: en cuanto a la facilidad de uso (lenguaje de interrogación), rapidez en la recuperación de los componentes, diversos tipos de acceso al repositorio, flexible y adaptable a los requerimientos de los usuarios (número de items recuperados, tipo de orden, localización, cantidad de información ofrecida de cada componente, etc.), posibilidad de realizar búsquedas sobre conjuntos de componentes previamente obtenidos y no sobre todo el repositorio (refinamiento), capacidad de combinar la interrogación directa (consultas) con capacidades de navegación hipertexto, etc.

Cada técnica de clasificación/recuperación presenta ventajas y desventajas con respecto a diversos criterios al compararse con las demás. En este artículo hemos tomado un subconjunto de los criterios mencionados en los párrafos anteriores y las técnicas descritas, asociando a cada técnica un valor para cada criterio, sin embargo, debemos advertir que esta asignación no es objetiva, en el sentido de que no se han llevado a cabo experiencias controlables, la valoración se hace en base a los comentarios que se ofrecen en la literatura, opiniones de diversos expertos en el área y experiencias del grupo de investigación ISRI, proyecto de reutilización coordinado por la Universidad de Vigo [ISRI98].

Técnica/Criterio	**Precision**	**Recuperación**	**Interface**	**Algorit. búsq.**	**Navegación**	**Aprend.User**
Indización	Media	Grande	Simple	Simple	No	Sencillo
Indiz. Automática	Media	Grande	Simple	Simple	No	Trivial
Facetas	Alta	Media	Media	Complejo	No	Medio
Marcos Conoc.	Alta	Media	Compleja	Medio	No	Complejo
Especif. Formales	Muy Alta	Pequeña	Compleja	Medio	No	Complejo
Redes Neuronales	Alta	Media	Compleja	Complejo	Si	Complejo
Hojear	Baja	Grande	Simple	Simple	Si	Trivial
Hipertexto	Creciente	Dependiente	Trivial	Trivial	Si	Trivial



5.Conclusiones

En el artículo se han expuesto las principales técnicas de clasificación/recuperación de componentes software reutilizables, identificando sus características, problemas y posibles soluciones, así como su previsible impacto en la calidad de los sistemas desarrollados mediante la composición de los componentes recuperados por estas técnicas. No existe, actualmente, ninguna técnica que, a nuestro parecer, resuelva el problema planteado: "almacenar componentes software en un repositorio y permitir su efectiva recuperación para usarlos en el desarrollo de nuevas aplicaciones". Sin embargo, ha de reconocerse que el trabajo de investigación en esta área es importante y que, gracias a los avances -tanto software como hardware- en diferentes técnicas, metodologías, y sistemas de almacenamiento, la solución parece estar próxima. En los próximos años, avances en la adquisición, representación y organización del conocimiento; tratamiento del lenguaje natural; tecnologías multimedia; repositorios distribuidos; motores de búsqueda; agentes inteligentes; comunicaciones -fundamentalmente INTERNET-, etc., nos ofrecerán alternativas importantes para la reutilización efectiva de componentes software. En cuanto a la Calidad, existe común acuerdo en el ámbito científico y empresarial que la reutilización de software produce importantes incrementos en los sistemas desarrollados, en buena parte el aseguramiento de esta calidad depende de los propios componentes, pero también de las técnicas usadas para recuperarlos, pués éstas pueden influir en la decisión de cual componente utilizar, es decir, debemos exigir a la técnica de clasificación/recuperación que "siempre" nos proporcione el "mejor componente".

6. Referencias

[Allan95] J. Allan: "Automatic Hypertext Construction", TR95-1484, Department of Computer Science, Cornell University, Ithaca, Feb. 1995.

[Allen89] B. Allen, S. Lee: "A Knowledge-Based Environment for the Development of Software Parts Composition Systems", Proc. 11th Int. Conference on Software Engineering, pp. 104-112, 1989.

[Asset] ASSET - Asset Source for Software Engineering Technology, http://source.asset.com/

[Borst92] J. Borstler: "Feature-Oriented Classification and Reuse in IPSEN", University of Technology AIB Nr. 92-35, Aachen, Germany, 1992.

[Borst93] J. Borstler: "FOCS: A Classification System for Software Reuse", Proc. of the 11th Pacific Northwest Software Quality Conference, Portland, OR, USA, pp. 201-211, October, 1993.

[Burto87] B. Burton, R. Aragon, S. Bailey, K. Koehler, and L. Mayes, "The Reusable Software Library", IEEE Software, Vol. 4, N° 4, pp. 25-33, July, 1987.

[Casta94] S. Castano and V. De Antonellis: The F^3 reuse environment for requirements engineering, ACM SIGSOFT Software Engineering Notes, Vol.19, N° 3, pp. 62-64, July, 1994.

[Covin94]M.A. Covington: "Natural Language Processing for Prolog Programmers", Prentice-Hall, Englewood Cliffs, 1994.

[Croft93] B. Croft: "Knowledge-based and Statistical Approaches to Text Retrieval", IEEE Expert, Vol. 8, N° 2, pp. 8-22, April, 1993.

[Cybul93]J.L. Cybulski, K. Reed: "The Use of Templates and Restricted English in Structuring and Analysis of Informal Requirenment Specifications", AAITP Technical Report TR024, 1993.

[DeMey93] V. de Mey, O. Nierstrasz: "The ITHACA Application Development", Visual Objects, D. Tsichritzis (Ed.), University of Geneva - Centre Universitaire d'Informatique, pp. 265-278, 1993.

[Devan91] P. Devanbu, R. Brachman, P.Selfridge, B. Ballard: "LaSSIE: A knowledge-based Software Information System", CACM, Vol. 34, N° 5, pp. 34-49, May, 1991.

[Eichm92] D. Eichmann, K. Srinivas: "Neural Network-Based Retrieval from Software Reuse Repositories", Neural Networks and Pattern Recognition in Human Computer Interaction, R. Beale and J. Findlay (Eds.), Ellis Horwood Ltd, West Sussex, UK, pp. 215-228, 1992.

[Fagan89] J.L. Fagan: "The effectiveness of a nonsyntactic approach to automatic phrase indexing for document retrieval", Journal of the American Society for Information Science, Vol. 40, N° 2, pp. 115-139, Sept., 1989.

[Fitzg88] J. Fitzgerald, R. Mathis: "Use of an Expert System in Software Component Reuse", Proc. AIDA'88, George Mason University, Fairfax, VA USA, 1988.

[Frake90] W.B. Frakes, T. Pole: "Proteus: A Software Reuse Library System that Supports Multiple Representation Methods", SIGIR Forum, Vol. 24, N° 3, pp. 43-55, 1990.

[Frake92] W.B. Frakes and E.R. Baeza-Yates (Eds.): Information Retrieval - Data Structures & Algorithms, Prentice Hall, Englewood Cliffs, 1992..

[Girar92] M.R. Girardi, R.T. Pierce: "A Toll for Software Reuse in Object-Oriented Development", V Brazilian Competition of Theses and Dissertations, published in Proc. XII Brazilian Computer Society Conference, Brazilian Computer Society, Rio de Janeiro, pp. 344-350, Oct., 1992.

[Goldb84] A. Goldberg: "Smalltalk-80: The interactive Programming Environment", Addison-Wesley, 1984.

[Helm91] R. Helm, Y.S. Maarek: "Integrating Information Retrieval and Domain Specific Approaches for Browsing and Retrieval in Object-Oriented Class Libraries", Proc. OOPSLA/ECOOP'91, ACM SIGPLAN Notices, Vol. 26, N° 11, pp. 47-61, Nov., 1991.

[ISRI98] http://www.uvigo.es/webs/proyectos/webisri

[Jeng93] J. Jeng, B. Cheng: "Using Formal Methods to Construct a Software Component Library", Proc. 4th European Software Engineering Conference (ESEC'93), I. Sommerville and M. Paul (Eds.), LNCS 717, Springer-Verlag, pp. 397-417, 1993.

[Karls96] Software Reuse: A Holistic Approach. Even-André Karlsson (Ed.), pp. 287-375. 1996.

[Knigh92] J. Knight, D.M. Kienzle: "Reuse of Specifications", Proc. 5th Workshop on Software Reuse, 1992.

[Lanca93]F.W. Lancaster, A.J. Warner: "Information Retrieval Today", Information Resources Press, 1993.

[Mitte94] R.T. Mittermeir: "Building a Repository of Software Components: A Formal Specifications Approach", Proc. 6th Workshop on Software Reuse (WISR6), 1994.

[Morel93]J. Morel, J. Faget: "The REBOOT Environment", Proc. of International Workshop on Software Reuse (ISRW-2), E. Guerrieri (Ed.), Lucca, Italy, March 24-26, 1993.

[Morga91] R. Morgan: "Component Library Retrieval Using Property Models", Proc. OOPSLA'86, ACM SIGPLAN Notices, Vol. 21, N° 11, The British Library, pp. 131-139, 1991.

[Neigh84]J.M. Neighbors: "The Draco Approach to Constructing Software from Reusable Components", IEEE Trans. on Software Engineering, Vol. 10, N° 5, pp. 64-74, New York, Sept., 1984.

[Nie93] J. Nie, F. Paradis, J. Vaucher: "Using Information Retrieval for Software Reuse", Proc. 5th Int. Conference on Computing and Information (ICCI'93), O. Abou-Rabia, C. Chang and W. Koczkodaj (Eds.), IEEE Computer Society, Sudbury, Ontario-Canada, pp. 448-452, May 27-29, 1993.

[Oster92] E. Ostertag, J. Hendler, R. Prieto-Diaz, C. Braun: Computing similarity in a reuse library system: an AI-based approach, ACL TOSEM, Vol. 1, N° 3, pp. 205-228, (July 1992)

[Podgu92] A. Podgurski, L. Pierce: "Behavior Sampling: A Technique for Automated Retrieval of Reusable Components", Proc. 14th Int. Conference on Software Engineering, Australia, pp. 349-357, May 1992.

[Priet91] R. Prieto-Díaz: Implementing faceted classification for software reuse, Comm. of the ACM, Vol. 34, N° 5,pp. 88-97, May, 1991.

[Rijsb80] C.J. van Rijsbergen: "Iformation Retrieval", Butterworths, 1980.

[Rolli91] E.J. Rollins, J.M. Wing: "Specifications as search keys for software libraries", Proc. Int. Conference on Logic Programming, K. Furukawa (Ed.), pp. 173-187, 1991.

[Salto71] G. Salton: "The SMART Retrieval System: Experiments in Automatic Document Processing", Prentice-Hall, Englewood Cliffs, 1971.

[Salto83] G. Salton, M. Gill: "Introduction to Modern Information Retrieval", McGraw-Hill, New York, 1983.

[Salto89] G. Salton: Automatic Text Processing - The Transformation, Analysis and Retrieval of Information by Computer, Addison-Wesley, 1989.

[Salto90] G. Salton, Z. Zhao, C. Buckley: "A Simple Syntactic Approach for the Generation of Indexing Phrases", TR 90-1137, Department of Computer Science, Cornell University, July, 1990.

[Salto94] G. Salton, Allan, C. Buckley, Singhal: "Automatic Analysis, Theme Generation and Summarization of Machine-Readable Texts", Science, Vol. 264, June, 1994.

[Savoy93] J. Savoy: "Effectiveness of Information Retrieval Systems Used in a Hypertext Environment", Hipermedia, Vol. 5, N° 1, 1993.

[Sindre93] G. Sindre, E. Karlsson: "A Method for Software Reuse through Large Component Libraries", Proc. of ICCI'93, O. Abou-Rabia, C. Chang, W. Koczkodaj (Eds.), pp. 464-468, May 27-29, 1993.

[Smeat92] A.F. Smeaton: "Progress in the Application of Natural Language Processing to Information Retrieval Tasks", The Computer Journal, Vol 35, N° 3, pp. 268-278, June, 1992.

[Somme90] I. Sommerville: "DSCC components catalog tool for software reuse", IEEE Software, Vol. 7, N° 3, May, 1990.

[Steel94] L. Steels, G. Schreiber, W. van de Velde (Eds.): "A FUTURE for knowledge acquisition", Proc. of EKAW'94, LNAI, Vol. 867, Springer, Berlin, 1994.

[Steig92] R.A. Steigerwald: "Reusable Component Retrieval with Formal Specifications", Proc. 5th Workshop on Software Reuse (WISR5), 1992.

[Tichy89] W. Tichy, R. Adams, L. Holter: "NLH/E: A natural language help system", Proc. 11th ICSE, Pittsburgh, pp.364-374, May, 1989.