Un Acercamiento a la Reutilización en Ingeniería de Software

Raquel ▪ Anaya de ▪ Páez

El reuso es por defecto la estrategia de resolución de problemas en la mayoría de las actividades del ser humano. Los científicos cognitivos coinciden en afirmar que lo primero que hacemos cuando enfrentamos un problema es determinar si éste ha sido resuelto antes; en caso contrario, buscamos nuestro espacio mental problemas análogos que ya se han resuelto y adaptamos su solución al problema actual; cuando ninguna de las anteriores situaciones se cumple, utilizamos entonces las habilidades y el conocimiento general analítico para resolución de problemas (Mili, 1995). Se puede hablar de reuso en

Raquel Anaya de Páez. Profesora del Departamento de Informática y Sistemas, Universidad EAFIT. Aspirante a doctorado en Ingeniería de Software de la Universidad Politécnica de Valencia, España. Email: Ranaya@sigma.eafit.edu.co

diferentes grados: reuso informal e intuitivo que se basa en el conocimiento individual de las personas, reuso esquematizado por medio de la utilización de procedimientos que guían el desarrollo de actividades y reuso orientado a productos cuando la reutilización se concreta en artefactos que representan el conocimiento y que facilitan la labor de reutilizar.

En el campo de la ingeniería de software el reuso ofrece un gran potencial en términos de productividad y calidad del software. *Productividad,* porque amplifica la capacidad de programación, en el sentido de escribir menos código, reduce la cantidad de documentación y pruebas que se deben realizar y genera un efecto de sinergía sobre la funcionalidad del sistema completo a partir de la funcionalidad de sus componentes. *Calidad,* porque el diseño de componentes se realiza pensando en su posterior utilización, con una documentación precisa, con procesos certificados de prueba y validación y con una estructuración adecuada de las partes del componente para que sea entendible por el usuario.

En el campo de la ingeniería de software el reuso ofrece un gran potencial en términos de productividad y calidad del software. *Productividad,* porque amplifica la capacidad de programación, en el sentido de escribir menos código, reduce la cantidad de documentación y pruebas que se deben realizar y genera un efecto de sinergía sobre la funcionalidad del sistema completo a partir de la funcionalidad de sus componentes. *Calidad,* porque el diseño de componentes se realiza pensando en su posterior utilización, con una documentación precisa, con procesos certificados de prueba y validación y con una estructuración adecuada de las partes del componente para que sea entendible por el usuario.

La demanda de sistemas de información cada vez más complejos, la dinámica de cambio en las organizaciones y la globalización de las operaciones del negocio bajo unas restricciones de tiempo, costo y calidad, hacen de la reutilización, una necesidad imperiosa dentro de todo proceso de ingeniería de software. En este contexto la reutilización puede ser definida como la propiedad de utilizar conocimiento, procesos, metododologías o componentes de software ya existente para adaptarlo a una nueva necesidad, incrementando significativamente la calidad y productividad del desarrollo (Frakes, 1996).

Este trabajo realiza un acercamiento introductorio al aspecto de reutilización en el campo de la ingeniería de software. En la primera parte se ofrece una visión histórica del desarrollo de la reutilización en el área; a continuación se presentan los principales enfoques, aproximaciones y niveles de abstracción de los objetos de reuso; la tercera parte describe el ambiente de desarrollo de software orientado al reuso, los pasos generales que sigue el proceso de reutilización e introduce las consideraciones de la organización para implantar la reutilización como una práctica estándar en el proceso de desarrollo; por último se presentan las conclusiones.

1. ORÍGENES

La reutilización aparece a finales de la década del 60 como una alternativa para superar la crisis del software (Kueger, 1992; Prieto, 1993). En una conferencia titulada *"Mass produced software components"*

realizada en 1968, McIlory introduce el concepto de reusabilidad y propone una librería de componentes de código fuente y técnicas automatizadas para su adaptación en diferentes grados de precisión y robustez. A partir de esta propuesta surgen soluciones de reutilización para tareas específicas en diversas áreas de la computación como conversión de entrada/salida, computación numérica y procesamiento y almacenamiento de texto en línea. La idea de McIlory fue clave para el concepto de *factoría de software*; este término fue acuñado por SDC (*System Development Corp*) en 1974; ellos establecieron un conjunto de procedimientos bien definidos para un proceso de software consistente y repetible, donde los programadores reutilizaban segmentos de código de desarrollos anteriores. Aunque el concepto de reuso no se encontraba definido dentro del proceso, la experiencia de la SDC fue el punto de partida para las futuras factorías de software, especialmente en Japón, donde el proceso de reuso formal fue definido e integrado más tarde.

A mediados de la década del 70, Robert Lanergan inició un proyecto de reuso que ahora es considerado como el primer caso de reuso formal en una organización. A finales de los años 70 se iniciaron los trabajos e investigaciones a nivel académico. En el primer *Workshop on Reusability in Programming* (1983), hubo un consenso de los investigadores acerca de la necesidad de extender la reutilización a diferentes niveles de abstracción. En 1980 la creación de programas de reuso a grande escala contribuyeron significativamente al desarrollo del área. A finales de esta década se intensificaron los trabajos de reuso y se hicieron importantes avances en técnicas de clasificación, sistemas de librerías, creación y distribución de componentes reusables y ambientes de soporte al reuso. En 1987 Freeman identifica el reuso del conocimiento del contexto como uno de los principales objetivos de investigación en reuso de software; en esta misma línea, Vic Basili extendió la definición de reuso para incluír el 'uso de cualquier cosa asociada con el proyecto de software, incluyendo conocimiento'. Esta nueva visión, abrió tendencias de investigación en otras áreas y ha contribuido a reconocer que el problema del reuso debe abordarse desde una confluencia de disciplinas. Los trabajos recientes han estudiado aspectos no técnicos del reuso desde el punto de vista administrativo, económico, cultural y legal. Hoy en día, el interés esta centrado en la integración adecuada de todos estos factores con el objetivo de institucionalizar el reuso como una práctica sistémica dentro de las organizaciones.

2. ENFOQUES DE LA REUTILIZACIÓN

Son variadas las aproximaciones, enfoques y consideraciones que diferentes autores dan al tema de la reutilización. La **tabla 1** presenta una visión general del tratamiento de la reutilización considerando diferentes aspectos (Frakes, 1996).

Los trabajos recientes han estudiado aspectos no técnicos del reuso desde el punto de vista administrativo, económico, cultural y legal. Hoy en día, el interés está centrado en la integración adecuada de todos estos factores con el objetivo de institucionalizar el reuso como una práctica sistémica dentro de las organizaciones.

TABLA 1
Visión General de la Reutilización

FACETA	TIPO	DESCRIPCIÓN
Aproximación tecnológica	Por generación	Parte de una especificación del problema que va siendo refinada en iteraciones sucesivas.
	Por composición	Utiliza pequeñas partes de software como invariantes para proveer funcionalidad variante por medio de su integración.
Enfoque metodológico	Desarrollo para reutilizar	Centrado en el desarrollo de componentes adecuados para ser utilizados en un problema específico - visión proveedor.
	Desarrollo con reutilización	Centrado en el proceso mismo de construcción de soluciones software a partir de componentes almacenadas en una biblioteca - visión demanda.
Modificación	Caja Blanca	Componentes que son reutilizados por modificación y adaptación.
	Caja Negra	Componentes que son reutilizados sin ninguna modificación.
	Adaptativo	Utiliza estructuras grandes de software como invariantes y restringe la variabilidad a un conjunto de argumentos o parámetros.
Alcance del desarrollo	Interno	Mide el nivel de reutilización de componentes de un repositorio interno o de componentes del mismo proyecto.
	Externo	Mide el nivel de reutilización de componentes que provienen de repositorios externos o la proporción de productos que fueron adquiridos.
Alcance dominio	Vertical	Reutilización dentro del mismo dominio de aplicación.
	Horizontal	Reutilización dentro de dominios de aplicación diferentes.
Objeto de reuso	Según el estado del proceso de desarrollo	Según la etapa en el ciclo de vida en la cual el conocimiento es producido o reutilizado. Tecnologías de reuso de amplio espectro y de limitado espectro (Biggerstaff).
	Según el nivel de abstracción	Según la propiedad del componente de representar conceptos abstractos, concretos o implementados. Código reciclado, componentes de código, esquemas (Frakes). Código fuente, diseño, especificación, objetos, texto, arquitecturas (Prieto-Díaz).
	Según la naturaleza del conocimiento	Según el tipo de conocimiento que se reutiliza concretado en artefactos o habilidades. Artefactos reutilizables: datos, arquitecturas, diseño, programas (Jones) Naturaleza del conocimiento: informal, esquematizado, herramientas y productos (Basili).

La *aproximación tecnológica* se refiere a la técnica utilizada para desarrollar componentes. El *enfoque metodológico* representa la manera como se integra la reutilización dentro del proceso mismo de desarrollo. La *modificación* sugiere la forma como los componentes son accedidos y manipulados para adecuarlos a nuevas necesidades. El *alcance de desarrollo* indica si la reutilización sólo se aplica a componentes internos o permite integrar componentes de otras bibliotecas. El *alcance del dominio* delimita la reutilización a una familia de sistemas o permite su aplicación entre diferentes dominios de aplicación. El *objeto de reuso* se refiere a los diferentes enfoques que diversos autores proponen al considerar la reutilización: desde el punto de vista del proceso de desarrollo, considerando el nivel de abstracción del componente o según el tipo de conocimiento que se reutiliza. Las consideraciones más relevantes serán tratadas en los siguientes apartados.

2.1 REUTILIZACIÓN GENERATIVA VS REUTILIZACIÓN COMPOSICIONAL

Desde el punto de vista tecnológico se distinguen dos categorías generales de reutilización: por composición y por generación. La aproximación *por composición*, se orienta a reusar productos; enfatiza la creación de nuevo software a partir de componentes almacenados en bibliotecas de componentes. En esta aproximación se pueden distinguir tres formas diferentes de manipular los componentes: Por *Caja blanca*, cuando se puede observar al interior del componente con el propósito de entenderlo y adecuarlo; generalmente se presenta en forma de librerías en las que el código fuente puede ser accedido y modificado. Por *Caja negra*, cuando el componente es reutilizado sin que pueda ser modificado. *Adaptativo* combina los dos enfoques anteriores puesto que permite observar algunas propiedades del componente y realizar algunas adecuaciones de éste con el propósito de adaptarlo a una nueva necesidad. Mientras que en la técnica de caja blanca y adaptativo es importante facilitar la parametrización y flexibilidad de adaptación del componente, en la técnica por caja negra es importante la verificación y certificación que demuestre su calidad (Prieto-Díaz, 1993).

La aproximación *por generación* se orienta en reutilizar el proceso de esfuerzos previos de desarrollo de software, generalmente concretados en herramientas que automatizan parte del trabajo de seleccionar, adecuar y generar nuevos componentes (Mili, 1995). Al igual que un proceso tradicional de desarrollo, el proceso de desarrollo del componente puede ser visto como una secuencia de transformaciones y/o traducciones de la descripción del problema en un lenguaje (nivel i) a otro (nivel $i+1$). Se distinguen tres niveles de conocimiento: acerca del dominio fuente (nivel i), acerca del dominio objetivo (nivel $i+1$) y acerca de cómo los objetos (entidades, relaciones, estructuras) del dominio fuente, se corresponden a objetos en el dominio objetivo. La correspondencia del conocimiento consiste entonces en un conjunto de reglas de transformación del nivel i al nivel $i+1$ y un conjunto de derivaciones ya realizadas entre las instancias del problema del nivel i y las instancias del problema del nivel $i+1$. Las reglas de transformación corresponden al proceso de reuso o habilidad de reuso, que se denomina la *gramática transformacional*.

El proceso consiste en describir el problema en el lenguaje del nivel *i* para obtener una descripción que luego se transforma en una descripción del siguiente nivel, utilizando el conocimiento involucrado en el proceso de transformación. Con el reuso se pretende realizar este proceso de transformación de manera asistida ya sea porque no se hace necesaria una especificación completa del problema (reuso a nivel de requerimientos) o porque las transformaciones de la especificación completa del nivel *i* son realizadas en forma automática en el nivel *i+1*. Este enfoque transformacional da lugar a diversos tipos de herramientas como los generadores de código que crean programas dada una especificación parametrizada o programada y los lenguajes de especificaciones de alto nivel que permiten especificaciones de las entidades u operaciones del dominio del problema directamente en la sintaxis del lenguaje.

Las aproximaciones generativa y composicional generalmente son combinadas para efectos prácticos, dando lugar a entornos de desarrollo que integran de manera homogénea la reutilización en las diferentes etapas del ciclo de vida.

2.2 PARA REUTILIZAR Vs CON REUTILIZACIÓN

Desde el punto de vista metodológico las tareas de reutilización se orientan desde dos aspectos que Moore llama la *visión proveedor* y la *visión demanda* y que generalmente se conoce como desarrollo *para* reutilizar y desarrollo *con* reutilización (Moore, 1991) (Mili, 1995). La visión proveedor, enfatiza la necesidad de adquirir (adecuar, comprar, desarrollar) componentes con la suficiente calidad para que puedan ser incluídos en la biblioteca de componentes para reutilizar; la visión demanda, destaca la necesidad de utilizar conocimiento que existe para satisfacer los requerimientos de un nuevo sistema.

Caldiera propone una estructura organizacional donde separa actividades relacionadas con el desarrollo del proyecto y actividades relacionadas con el manejo de una biblioteca de componentes (Caldiera, 1991). El grupo del proyecto construye soluciones específicas de software a partir de componentes que ya existen. El equipo encargado de la biblioteca de componentes, puede interactuar de manera sincrónica o asincrónica con el grupo del proyecto de desarrollo. En el trabajo sincrónico, los expertos en el desarrollo de componentes atienden solicitudes del equipo del proyecto para desarrollar aquellos que satisfagan una necesidad concreta del sistema en desarrollo. En el trabajo asincrónico, los expertos de la biblioteca desarrollan un plan de producción de componentes que representan oportunidades de reuso dentro de un dominio.

Los ambientes y herramientas que soportan el desarrollo de componentes para reutilizar se ocupan de tres actividades principales: *Análisis del dominio*, que generaliza las características comunes de sistemas similares. Este proceso permite identificar objetos, operaciones y relaciones que los expertos del dominio perciben que son importantes acerca del dominio (Arango, 1991). *Desarrollo y refinamiento de productos reusables*, donde se desarrollan los componentes que posteriormente van a ser reutilizados o donde se refinan o adecuan productos que existen para

aumentar su reusabilidad. *Clasificación*, donde se identifican características comunes a más de un componente; de la manera como los componentes sean clasificados depende en alto grado la flexibilidad y rendimiento de los procesos de búsqueda y recuperación para obtener componentes candidatos a reutilizar.

Un dominio de aplicación es el área de conocimiento hacia la cual se orienta determinada solución de software. El reuso del conocimiento del dominio de aplicaciones bajo la forma de *modelos del dominio* es un área que ha recibido considerable atención y es de vital importancia para el tratamiento de la reutilización a alto nivel de abstracción (Prieto-Díaz, 1991). El proceso para obtener este conocimiento es denominado el *análisis del dominio* y el resultado de este análisis recibe el nombre de *modelo de dominio* o *lenguaje del dominio* (Neighbors, 1989) (Hall, 1992). Este modelo captura el significado de los conceptos manejados por el usuario y sus relaciones semánticas con otros conceptos y tiene por objetivo: Ayudar al desarrollador a entender un dominio de aplicación, servir como punto de partida para el análisis del sistema y proveer una categorización/clasificación de componentes existentes reusables de tal manera que las oportunidades para reutilizar puedan ser identificadas tan temprano como sea posible dentro del proceso de desarrollo de software (Mili, 1995).

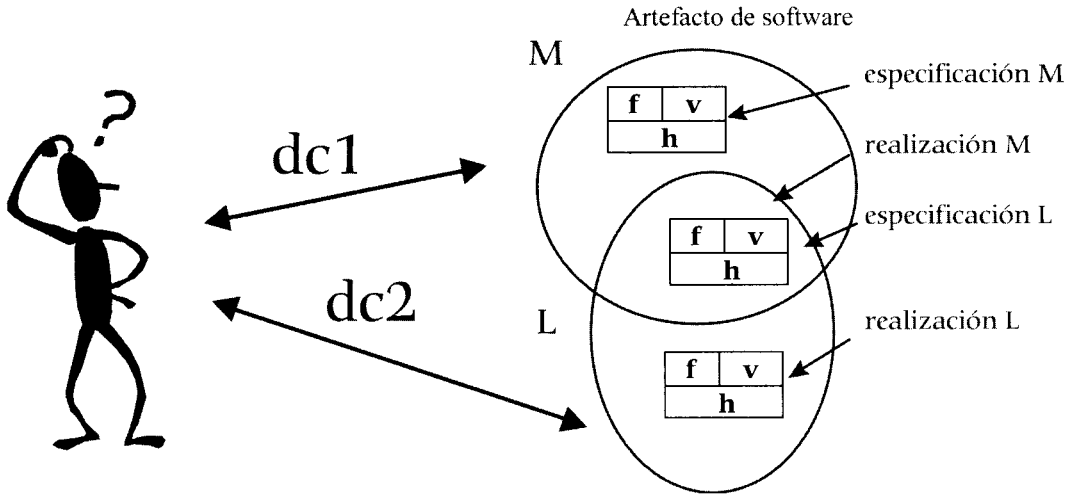
En el desarrollo con reutilización se enfatiza la necesidad de encontrar componentes almacenados que satisfagan un conjunto dado de requerimientos. Los pasos generales que sigue este proceso son: *encontrar* el componente, *entenderlo*, *modificarlo* y *componerlo* 'para generar nuevos componentes. Para encontrar el componente se requieren métodos automáticos que asistan a los desarrolladores en su búsqueda y recuperación. El entendimiento del componente involucra diferentes grados de conocimiento del componente, determinados básicamente por el tipo de tratamiento que se le aplica (caja blanca, caja negra, adaptativo): primero, entender *qué* hace; segundo, entender *cómo* lo hace y tercero, entender *cómo modificarlo* de tal manera que haga algunas cosas de manera diferente.

2.3 OBJETO DE LA REUTILIZACIÓN

Este trabajo hace énfasis en el objeto de reuso desde la perspectiva del nivel de abstracción del componente. Lo que diferencia un tipo de componente de otro es su nivel de abstracción. En términos generales cuanto más alto sea el nivel de abstracción, la potencialidad para entender, redefinir y adaptar será mayor. Es importante entonces considerar la efectividad de la abstracción en una técnica de reutilización de software. Krueger evalúa esta característica en términos del esfuerzo intelectual requerido para utilizar el componente (Krueger, 1992). Las mejores abstracciones significan que el usuario requiere menos esfuerzo para su utilización. La *distancia cognitiva* es definida como la cantidad de esfuerzo intelectual realizado por los desarrolladores para tomar un componente de software en un estado y llevarlo a otro estado; aunque no es una medida formal que puede ser expresada en números y unidades da una noción intuitiva del esfuerzo de reutilizar.

Toda abstracción de software tiene dos niveles (**figura 1**). El nivel más alto se refiere a

FIGURA 1
El Principio de Abstracción



la especificación de la abstracción y el nivel más bajo se refiere a la realización de la abstracción, que no necesariamente se trata del código fuente. A su vez, una abstracción debe estar conformada por una parte *oculta (h)*, una parte *variable (v)* y una parte *fija (f)*. La parte *oculta* contiene detalles de realización de la abstracción que no son visibles en el nivel de especificación de la abstracción; la parte *variable* representa las características modificables en la realización de la abstracción y la parte *fija* representa las propiedades de la abstracción que determinan su comportamiento básico. En la **figura 1**, se tiene un artefacto X con dos abstracciones M y L, con su respectiva distancia cognitiva (dc1, dc2). Si la abstracción M es por ejemplo la especificación estructural de una venta escrita en un modelo Entidad-Relación, la realización de esta abstracción será el esquema lógico de la base de datos que será a la vez la especificación L del siguiente nivel; la realización de L será, por ejemplo, una representación en estructuras de registros del esquema lógico. A continuación se describen los principales objetos de reuso y su evaluación frente al esfuerzo requerido para su reutilización.

Código Reciclado

Es la aproximación primitiva de reutilización donde los programadores recuperan código fuente generalmente construido por ellos mismo en desarrollos anteriores con el propósito de reducir el esfuerzo de diseño, escritura y prueba del nuevo software. La reutilización está originada por la experiencia y capacidad del programador de recordar desarrollos anteriores que guardan analogía con el nuevo problema. En estos casos generalmente la persona que reutiliza es la misma que construye lo cual disminuye la distancia cognitiva.

Componentes de Código

Esta fue la noción inicial de reutilización que introdujo McIlory: componentes de código adquiridos con el propósito de ensamblarlos

para construir nuevo software. Estos componentes son escritos, evaluados y almacenados específicamente con el propósito de ser reutilizados. Los lenguajes de programación orientados a objetos impulsaron considerablemente este tipo de reutilización. Hoy en día, son innumerables las librerías que se distribuyen con una funcionalidad específica y modularizada por naturaleza a través de la definición de clases. Por el principio de encapsulación, la definición de la interfaz de los métodos (nombre, argumentos de entrada y argumentos de salida) que ofrece cada clase, permite un acercamiento gradual al código sin tener que conocer los detalles de implantación de cada método. En otras palabras, la definición de clase es un mecanismo de abstracción que separa las propiedades de la clase (especificación) de sus instancias (realización). Sin embargo, en este tipo de reutilización, las clases y métodos representan unidades de reuso a muy pequeña escala, originando un problema de composición para ensamblar una funcionalidad útil en el dominio del problema. La distancia cognitiva depende del grado familiaridad que el programador tiene de las bibliotecas de clase, algunas de las cuales no ofrecen una documentación adecuada.

Esquemas

Enfatiza la utilización de abstracciones para representar una solución software a un alto nivel de abstracción. El esquema describe un conjunto de especificaciones escritas bajo determinado formalismo (modelo entidad – relación, diagramas de flujos de datos, diagramas de clases, especificaciones formales etc.) que recibe el nombre de *modelo conceptuàl*. La reutilización de estos modelos permiten que el analista aproveche especificaciones ya existentes y validadas para la construcción de nuevas especificaciones. El alto nivel de abstracción de la especificación reduce considerablemente la distancia cognitiva entre el requerimiento y la implementación de algoritmos y estructuras de datos. La mayoría de productos y esfuerzos de investigación de reutilización a alto nivel se concentran en la reutilización de modelos conceptuales.

El trabajo de Almeyer et.al define modelos conceptuales en diferentes formalismos orientados a dominio específicos y un entorno de desarrollo que permite derivar y caracterizar nuevos modelos (Altmeyer, 1997). El trabajo de Ruggia et.al utiliza modelos Entidad - Relación Extendidos y provee un conjunto de herramientas que ofrecen servicios de reuso (Ruggia, 1997). Trabajos como el de Ambrosio (Ambrosio, 1996) y (Castaño, 1997) también utilizan el modelo Entidad-Relación como formalismo para representar una especificación e introducen además un modelo de métricas para medir la afinidad entre diferentes modelos conceptuales. Bajo el enfoque Orientado a Objetos uno de los trabajos más representativos es el proyecto Ithaca, que desarrolló un ambiente para reutilización de especificaciones a diferente nivel de abstracción, en el que utiliza un modelo de especificación OO con funcionalidad extendida llamado F-ORM (*Funcionality Object with Roles Model*). Este modelo enriquece el concepto de clase para manejar clases que representan comportamientos (clases de procesos) además de las clases que describen estructura y comportamiento (clases de recursos) (Bellinzona, 1994).

Frameworks

Una nueva tendencia para facilitar la reutilización de arquitecturas de software es el concepto de *framework*. Un *framework* es una solución integrada ejecutable de comunicación entre un conjunto de clases abstractas que representan un comportamiento útil en el espacio del problema. Los *framework* representa soluciones en un contexto particular y proveen mecanismos eficientes para la adecuación del comportamiento a una nueva necesidad. El trabajo de Baumer et.al, por ejemplo, presenta un *framework* para desarrollo de sistemas a grande escala en proyectos financieros (Baumer, 1997). Otros trabajos proponen el desarrollo de *framework* a partir de patrones de diseño. Los patrones de diseño ofrecen una terminología y comportamiento básico que permite capturar las especificaciones de un diseño reusable orientado a objetos (Pree, 1994) (Meijler, 1997).

En resumen, un alto nivel de abstracción en las técnicas de reutilización, reduce el esfuerzo requerido para ir del concepto inicial (requerimiento) a su representación en una especificación y automáticamente reduce el esfuerzo para ir de la abstracción a una implementación ejecutable.

3. AMBIENTE DE DESARROLLO DE SOFTWARE ORIENTADO AL REUSO

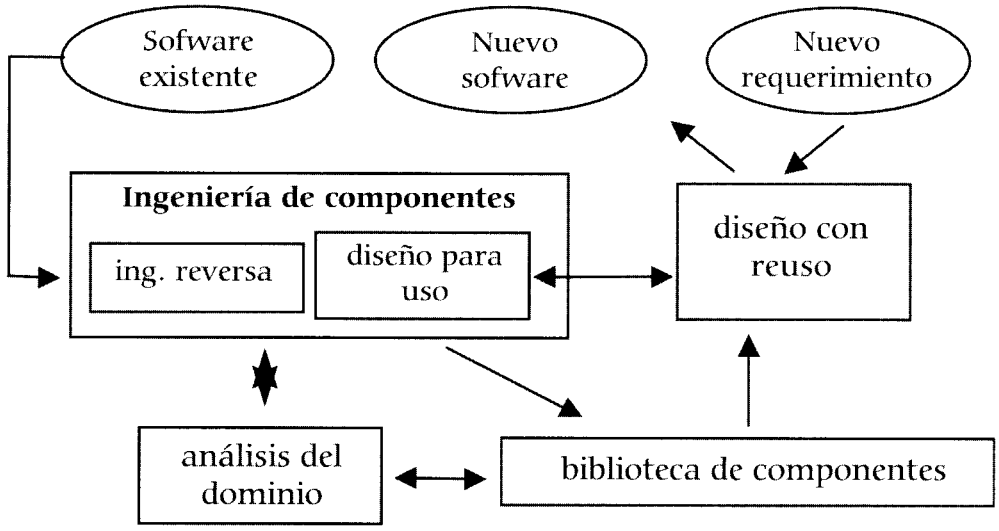
El objetivo es involucrar formalmente la reutilización como un disciplina e integrada en la metodología de desarrollo del software. Esto significa que las funciones típicas de reuso como búsqueda, extracción y edición de componentes no deben distraer a los analistas de su flujo normal de trabajo (Mili, 1995). Es decir, los entornos de desarrollo o ambientes CASE deben ser orientados al reuso, integrando de manera homogénea la funcionalidad requerida para reutilizar de tal manera que se conviertan en extensiones del espacio de trabajo mental del analista.

La **figura 2** ilustra la arquitectura básica de un ambiente de reutilización en el que se combinan el diseño *para* reuso y el diseño *con* reuso. La ingeniería de componentes permite manipular todos los objetos para almacenarlos en una librería de componentes de tal manera que puedan ser recuperados para su posterior utilización. El módulo de diseño donde los componentes de software son seleccionados a partir de un nuevo requerimiento. El análisis del dominio que permite identificar, interpretar y recuperar los objetos almacenados en la biblioteca de componentes. La biblioteca de componentes está conformada por un repositorio para almacenar componentes candidatos a reutilizar más una interfaz de búsqueda.

3.1 ETAPAS EN EL PROCESO DE REUTILIZACIÓN

Etapa 1. Adquisición del requerimiento. Es el paso inicial donde el analista especifica al sistema una necesidad de información. Idealmente esta necesidad debe estar formulada en el contexto del problema y no en contexto de la solución. Es decir, el analista no debe estar obligado a conocer el dominio de la solución. En estos casos el modelo del dominio se convierte en un modelo de requerimientos que facilita la labor de precisar un requerimiento.

FIGURA 2
Arquitectura de un Ambiente de Reutilización



Etapa 2: Búsqueda y Recuperación. Es el mecanismo de búsqueda que permite evaluar los objetos de la biblioteca para recuperar el subconjunto de objetos candidatos a reutilizar. El problema principal radica en cómo organizar y recuperar los diversos componentes de software que son potencialmente reutilizables de tal manera que cualquier desarrollador pueda accederlos de manera consistente y flexible. La mayoría de las técnicas de recuperación tienen su origen en las técnicas utilizadas para recuperación de elementos en una base de datos documental; las técnicas más importantes son:

Recuperación por palabras claves

También denominada recuperación basada en descriptores léxicos (Mill, 1995) o recuperación multifaceta (Prieto-Díaz, 1991). Cada componente de la biblioteca tiene asignado un conjunto de palabras claves tomadas de un vocabulario predefinido que establece los criterios de búsqueda relevantes. Se formula una consulta que contiene una o más de las palabras claves y el sistema se encarga de recuperar los objetos que tengan asociado este concepto. Una de las clasificaciones más conocidas es propuesta por Prieto-Díaz, que establece una clasificación en seis facetas con criterios como área funcional hacia la cual se orienta el componente, función que desempeña el componente dentro del modelo, etc. La tecnología de hipertexto puede ser utilizada con éxito como interfaz de búsqueda y recuperación de componentes clasificados por palabras claves (Isakowitz, 1996). Las principales desventajas de esta técnica son el alto costo de clasificación manual para todos los componentes de la biblioteca, la necesidad de personal entrenado para realizar la clasificación y la ambigüedad o desacuerdo que se presenta en cuanto al significado de los términos.

Recuperación basada en la estructura

En esta técnica los términos en que se describe la consulta dependen de la semántica del componente. El lenguaje para describir la consulta y el lenguaje para especificar el componente, deben ser idénticos o debe existir una correspondencia entre ellos (Mili, 1995). Este tipo de recuperación generalmente ofrece al usuario una plantilla que permite describir las propiedades estructurales del componente. Se distinguen dos métodos básicos de recuperación: *Recuperación basada en orden parcial,* donde se establece un orden de relevancia de las propiedades a comparar con el propósito de reducir el número de comparaciones entre los componentes. *Recuperación basada en distancia*, donde los métodos introducen algún modelo de métricas de distancia para encontrar la afinidad entre los componentes. Esta aproximación es principalmente utilizada en la etapa de identificación del componente.

Recuperación enumerada

También denominada *recuperación basada en taxonomía* (Mili, 1995). Permite recorrer los conceptos en un orden determinado, generalmente según su ubicación en una jerarquía de herencia. Permite al usuario familiarizarse con los términos, conceptos o componentes de un dominio del problema a diferentes niveles de abstracción.

Un entorno de desarrollo orientado a la reutilización, debe integrar adecuadamente más de un método de recuperación. Aunque cada uno de estos métodos presenté una interfaz diferente al usuario, todos estos mecanismos pueden derivar expresiones de consulta a la biblioteca de componentes en un único formalismo.

Etapa 3: Identificación. En esta etapa se examinan los objetos recuperados para seleccionar aquellos que posean la funcionalidad deseada. Aquí es deseable contar con un sistema de métricas que mida la afinidad entre los componentes del esquema. Es decir, la métrica de similaridad se convierte en una herramienta que ayuda al diseñador para identificar los componentes adecuados a un requerimiento. La identificación es un paso posterior que complementa la recuperación de la etapa anterior. Mientras que la recuperación (*retrieve*) permite obtener de la biblioteca un conjunto de componentes utilizando un criterio general de selección, la identificación (*browsing*) es un proceso más detallado que explora los componentes recuperados teniendo un componente como referencia (*current object*) y generando una vista del vecindario de interés, generalmente con alguna clase de medida de distancia (Constantopoulos, 1997)(Motro, 1995).

Etapa 4: Adecuación. En esta etapa se debe disponer de herramientas para abstraer componentes afines y adecuarlos a su nuevas necesidades. Las operaciones de adecuación pueden darse al interior de un modelo (intra-esquemas) o entre diferentes modelos (inter-esquema). El trabajo de Ruggia et. al, por ejemplo, define operaciones de unión, intersección, substracción y extracción entre los componentes de esquemas diferentes (Ruggia, 1997).

3.2 CONSIDERACIONES ORGANIZACIONALES

Es importante evaluar el nivel de reutilización como estrategia para mejorar la calidad del proceso de producción de software. Frakes hace referencia al Modelo de Madurez de la Reutilización en el cual se presentan diferentes niveles de reutilización en la organización, en aspectos como motivación del personal de sistemas, planeación de la reutilización, personal involucrado en el proceso, tecnología de soporte, métricas utilizadas y consideraciones legales (Frakes, 1996). La **tabla 2** presenta una visión simplificada de este modelo.

TABLA 2
Vista General del Modelo de Madurez de la Reutilización

Nivel	Características
Caótico	Aplicación individual intuitiva e informal de componentes que no están orientados al reuso.
Monitoreado	Aplicada a nivel de grupos de trabajo motivados por la iniciativa a compartir, que utilizan herramientas no especializadas para la reutilización.
Coordinado	Aplicada en algunas áreas de aplicación o proyectos con algunas herramientas de clasificación y síntesis.
Planeado	Orientado al desarrollo de familias de productos motivado por la naturaleza del negocio con librerías electrónicas de componentes reusables.
Integrado	Aplicada en toda la empresa como forma de trabajo, incentivada por el plan estratégico y con soporte automatizado orientado a la reutilización.

Para que la reutilización sea adoptada de manera integrada o sistémica se requiere apoyo logístico y un replanteamiento de las metodologías de desarrollo. Prieto-Díaz propone una infraestructura administrativa conformada por bibliotecarios o administradores de repositorios, grupos de calidad, ingenieros reutilizadores, diseñadores de componentes para reuso, etc. Se han realizado estudios empíricos para determinar las consideraciones organizacionales del proceso de adopción de la reutilización. Para que una empresa pueda adoptar la reutilización de manera sistémica debería concentrarse en aspectos como educar a los desarrolladores acerca de los principios de la reutilización, crear conciencia de la economía que implica reutilizar y dedicar esfuerzos y proponer estrategias para que la alta calidad se convierta en una premisa básica del desarrollador.

Frakes hace referencia al Modelo de Madurez de la Reutilización en el cual se presentan diferentes niveles de reutilización en la organización, en aspectos como motivación del personal de sistemas, planeación de la reutilización, personal involucrado en el proceso, tecnología de soporte, métricas utilizadas y consideraciones legales.

CONCLUSIONES

Se ha presentado una visión general de la reutilización en el área de ingeniería de software. La reutilización es una estrategia adecuada para mejorar la productividad y calidad de un producto software. Aunque generalmente se considera la reutilización de código fuente, la reutilización aplicada a lo largo del ciclo de vida, logrará mayores beneficios, puesto que el alto nivel de abstracción de los componentes a reutilizar, reducen el esfuerzo requerido de ir del concepto inicial (requerimiento) a su representación en un especificación y de ésta a su implantación ejecutable.

A pesar de las expectativas prometedoras, hoy en día, la reutilización no representa una práctica estándar en el proceso de desarrollo de software. En la mayoría de las organización ésta se realiza de manera intuitiva e informal dependiendo de la experiencia y habilidad del desarrollador y por una iniciativa personal. Esto ha motivado a los investigadores para estudiar cómo y dónde aplicar la reutilización y proponer mecanismos y estrategias que permitan su utilización en forma integrada y sistémica. Los entornos de desarrollo deben integrar de manera homogénea la reutilización como estrategia básica de trabajo de los desarrolladores. La organización debe evaluar su nivel de reutilización y debe trabajar para que ésta se convierta en una disciplina involucrada en el proceso mismo de desarrollo de software y cuente con la tecnología apropiada.

MATERIAL DE REFERENCIA

Altmeyer, Joachim y otros. (1997). Application of a Generator-Based Software Development Method Supporting Model Reuse. Advanced Information System Engineering. CAISE'97. Eds. Antoni Olivé, Joan Antoni Pastor. Barcelona España.

Ambrosio, Ana Paula. (1996). Applying similarity vectors for the selection of reusable schemas. Memorias XXII Conferencia Latinoamericana de Informática.

Arango, Guillermo. (1991). Domain Analysis - From Art Form to Engineering Discipline. En Domain Analysis and Software Systems Modeling. IEEE Computer Society Press.

Baumer, Dirk y otros. (1997). Framework Development for Large Systems. Comunications of the ACM. Octubre de 1997.

Bellinzona, Roberto; Fugini, Maria; Pernici, Barbara. (1995). Reusing Specifications in OO Applications. IEEE Software. Marzo de 1995.

Caldiera, G.; Basili, V. (1991). Identifying and qualifying reusable software components. Computer, vol. 24, no. 2.

Castaño, Silvana; De Antonellis, Valeria. (1997). Engineering a library of reusable conceptual components. Information and Software Technology Vol 39, no.2.

Constantopoulos, Panos; Pataki, Elena. (1992). A Browser for Software Reuse. Proceeding 4th

International Conference CAISE' 92. De. P. Loucopoulos.

Frakes, William; Fox, Christopher. (1995). Sixteen Questions About Software Reuse, ACM Comunications. Volumen 38 no. 6.

Frakes, William; Terry, Carol. (1996). Software Reuse: Metrics and Models, ACM Computing Surveys. Volumen 28 no. 2.

Hall, P.A.V. (1992). Overwiew of reverse engineering and reuse search. Information and Software Technology Vol 34, no. 4.

Isakowitz, Tomas. (1996). Kauffman, Robert J. Supporting search for reusable objects. Transactions on software engineering, Volumen 22 no. 6.

Krueger, Charles W. (1992). Software Reuse, ACM Computing Surveys. Volumen 24 no. 2, Junio de 1992

Meijler, Theo y otros. (1997). Making Design Patterns Explicit in Face. A Framework Adaptive Composition Environment. Software Engineering Notes. 6th European Software Engineering Conference, 5th ACM SOGSOFT Symposium. Eds. Mehdi

Mili, Hafedh; Mili, Fatma; Mili, Ali. (1995). Reusing Software: Issues and Research Directions. IEEE Transactions on Software Engineering. Junio de 1995.

Moore, John. (1991). Domain Analysis: Framework for Reuse. En Domain Analysis and Software Systems Modeling. IEEE Computer Society Press.

Motro, Amihai; Goullioud Sylvie. (1995). Knowledge organization for exploration. Memorias DEXA'95.

Neighbors, J.M. Draco. (1989). A Method for Engineering Reusable Software Systems. Software Reusability, Volumen I, ACM Press.

Pree, Wolfgang. (1994). Meta Patterns - A Means for Capturing the Essentials fo Reusable Object-Oriented Design. Object Oriented Programming. ECOOP'94 Proceedings. Ed. Springer Verlag.

Prieto-Díaz, Rubén. (1991). Implementing Faceted Classification for Software Reuse, Comunications of the ACM. Volumen 34 no. 5.

Prieto-Diaz, Rubén; Arango, Guillermo. (1991). Domain Analysis and Software Systems Modeling. IEEE Computer Society Press.

Prieto-Díaz, Rubén. (1993). Status Report: Software Reusability. IEEE Software. Mayo de 1993.

Ruggia, Raúl; Ambrosio, Ana Paula. (1997). A toolkit for Reuse in Conceptual Modelling. Proceeding Advanced Information System Engineering. CAISE'97. Eds. Antoni Olivé, Joan Antoni Pastor. Barcelona España.

Tracz, Will. (1990). Tutorial: Software Reuse: Emerging Technology. The Computer Society of IEEE.