ESPECIFICAÇÃO DE UMA FERRAMENTA DE APOIO A REUTILIZAÇÃO DE SOFTWARE NO DESENVOLVIMENTO ORIENTADO A OBJETOS

MARIA DEL ROSARIO GIRARDI
ROBERTO TOM PRICE

CPGCC/II - UFRGS
Caixa Postal 1501
90.001 - Porto Alegre - RS
bitnet: TOMPRICE@SBU.UFRGS.ANRS.BR
fax: (0512)24.4164

Resumo

É descrita uma ferramenta para o reuso de software em ambientes com suporte ao paradigma de desenvolvimento por objetos. A ferramenta consiste de um mecanismo de recuperação de classes, por busca indexada, de uma base de software, baseado em descritores de classe. São incorporadas técnicas simples de aprendizagem que incrementam e atualizam os descritores existentes, a partir do sucesso ou falha na recuperação, de forma a melhorar, progressivamente, a efetividade do mecanismo de recuperação. Um mecanismo de busca exploratória, através das hierarquias de classes, é proposto como alternativa complementar para recuperação de classes reutilizáveis.

Abstract

A tool for software reuse in object-oriented environments is described. The tool for class retrieval on a software base, implements an index search mechanism based on class descriptors. A simple learning facility enables the tool to increment and update the set of class descriptors, reflecting the sucess ou fail of each retrieval, in order to increase, in progress, the mechanism efficiency. Exploratory browsing through the class hierarchies is proposed as a complementary way of retrieving reusable classes.

1 Introdução

Os sistemas desenvolvidos segundo o paradigma de objetos [GIR90b, TAK90] são construídos como objetos complexos constituídos pela composição de outros objetos, cada um com propriedades e comportamento próprios. É possível que esses objetos sejam projetados especificamente para a aplicação em desenvolvimento. Entretanto, a principal potencialidade do paradigma, para o incremento da qualidade e produtividade de software, reside nas suas facilidades para reutilização de software, como instanciação ou especialização, por herança, de classes pré-definidas [GIR90b]. Por isso, idealmente, a maioria dos objetos que compõem as aplicações deveriam ser obtidos reutilizando classes de objetos de uso geral existentes numa base ou biblioteca de software.

As aplicações desenvolvidas a partir de um conjunto de classes pré-definidas, e portanto, já testadas, apresentarão um maior grau de confiabilidade. O custo de produção de cada classe projetada é dividido entre todas as aplicações que a utilizam, com redução considerável do custo por aplicação. A

manutenção, tanto corretiva como evolutiva é simplificada pelo uso de classes padronizadas e pela total modularidade da arquitetura dos sistemas orientados a objetos [GIR89].

O incremento da confiabilidade e manutenibilidade dos sistemas orientados a objetos e da produtividade de desenvolvimento depende, portanto, da existência de ambientes que forneçam mecanismos para criar classes de ampla aplicabilidade e mecanismos que permitam reutilizar essas classes em aplicações específicas. Esses mecanismos são bastante limitados nos ambientes existentes [GIR90c]. É necessário incorporá-los com metodologias [GIR89, ROT87, TSI89] e ferramentas [GIR90a] que facilitem as tarefas de construção, busca, recuperação e integração de classes, e permitam a construção de aplicações complexas, reutilizando classes disponíveis em bibliotecas.

Bibliotecas com um número pequeno de classes estáveis são assimiladas rapidamente pelos usuários, permitindo que estes selecionem classes e métodos diretamente pelos seus nomes ou buscando classes apropriadas navegando através da hierarquia de classes. Grandes bibliotecas, ou bibliotecas com conteúdo que evolue rapidamente, tornam difícil a recuperação de classes e a simples navegação através das hierarquias de classes pode transformar-se num processo muito trabalhoso que desestimula a reutilização das classes existentes. São necessários mecanismos de procura mais rápidos que permitam a recuperação de classes a partir da especificação de requisitos que descrevem a classe.

O mecanismo padrão para recuperar informação de um Banco de Dados é chamado de *recuperação sistemática* [MOT 86]: os requisitos do usuário são especificados numa consulta formal e o gerenciador do banco de dados recupera a informação rapidamente.

Há situações, entretanto, nas quais a recuperação sistemática é difícil ou impossível, como por exemplo, quando os critérios de seleção são vagos ou não podem ser explicitados satisfatoriamente. Nessas situações, a busca exploratória ou "browsing", através das hierarquias de classes, é uma boa alternativa. A procura começa numa localidade arbitrária e, a medida que progride, o usuário ganha conhecimento da natureza e organização do espaço de busca [PIN88, RAJ89].

Por outro lado, os mecanismos de recuperação deveriam utilizar o conhecimento de usos anteriores da biblioteca de forma a capturar a experiência dos usuários no processo de reuso.

Este artigo descreve uma ferramenta [GIR 90a] de apoio ao processo de recuperação de classes para sua reutilização no desenvolvimento de software segundo o paradigma de objetos. A ferramenta fornece um mecanismo de recuperação sistemática, utilizando a busca exploratória como mecanismo complementar. O conhecimento adquirido do processo de recuperação de classes, através dos sucessivos usos da ferramenta, é incorporado à biblioteca de forma a viabilizar, progressivamente, o sucesso do mecanismo de recuperação sistemática.

2 Premissas básicas

(a) A ferramenta proposta tem por objetivo auxiliar o processo de reuso no desenvolvimento de software segundo o

paradigma de objetos. Pretende-se incorporá-la num ambiente que suporte esse paradigma e integrá-la com as ferramentas fornecidas pelo ambiente.

(b) A proposta aborda o problema de recuperação das classes que melhor se adaptem aos requisitos especificados pelo usuário. São desconsideradas outras tarefas próprias do processo de reuso como o projeto e criação de novas classes (com capacidade de reuso), adaptação e integração das classes selecionadas, etc.

(c) A efetividade do mecanismo de recuperação depende, em grande parte, tanto da organização e estruturação da biblioteca de classes, como do conhecimento, nela armazenado, dos critérios através dos quais as classes poderiam ser identificadas para sua recuperação [ARA88a, EMB87].

A herança de classes permite definir uma relação estrutural para a biblioteca. Essa estrutura facilita a busca exploratória, fornecendo um guia para navegação em diferentes níveis de abstração, do geral ao particular e vice versa, através da hierarquia de classes. Por outro lado, para uma efetiva recuperação sistemática, é necessário estabelecer critérios de classificação e indexar as classes da biblioteca segundo descritores, através dos quais possam ser recuperadas posteriormente.

(d) A ferramenta deverá ter capacidade para gerenciar grandes bibliotecas, em continua evolução e com muitos grupos de classes similares. Em consequência, pretende-se especificar um mecanismo de recuperação eficiente, tanto em rapidez de recuperação, como na adequação das classes recuperadas aos requisitos especificados pelo usuário.

(e) A maioria dos sistemas com suporte ao reuso de software, propostos na literatura [PRI87, FRA87, SUG 86] promovem a reutilização de abstrações funcionais.

Os conceitos do paradigma de objetos, *objetos*, *classes* e *hierarquias*, constituem-se num outro mecanismo de abstração de componentes de software que conjuga abstração de dados com herança de classes. É promovido o reuso tanto da estrutura de dados como das funções do componente.

Considerando esses aspectos, a abordagem apresentada propõe a procura por uma classe para reusá-la como "caixa preta" (por simples instanciação) ou para especializá-la (através do mecanismo de herança).

Também é considerada a procura por um método determinado. Nesse caso, o sistema fornece as classes através das quais pode reutilizar-se o método procurado.

(f) Alguns estudos [PRI87, WOO88] indicam a frequente necessidade de modificar os componentes de software recuperados para adequá-los aos requisitos da aplicação. Por esse motivo, a ferramenta considera a recuperação tanto das classes que satisfazem perfeitamente os requisitos do usuário, como daquelas que satisfazem esses requisitos em forma parcial.

As classes recuperadas são apresentadas ordenadas segundo a relevância de cada classe em relação aos requisitos especificados. Ou seja, primeiro apresentam-se as classes que se adequam perfeitamente aos requisitos seguidas daquelas que os satisfazem em forma parcial e ordenadas segundo o grau crescente de parcialidade. Entre duas classes que têm a mesma relevância, o mecanismo de recuperação estabelece uma subordem baseada no número de seleções ou rejeições das classes segundo

esses descritores, em usos anteriores da ferramenta.

(g) Na compreensão de uma classe, é desejável que os detalhes secundários não ocultem os aspectos relevantes implementados na classe. Uma classe genérica possui menos detalhes de implementação que suas especializações e, por isso, é mais fácil de se entender. Por outro lado, consideramos mais simples adaptar uma classe a um novo contexto agregando-lhe novas características (ou restringindo-la) do que generalizando-la. Por isso, o sistema procura encontrar as *classes mais gerais*, na hierarquia de classes, que satisfazem os requisitos do usuário. Se o usuário procura por um método, o sistema seleciona as classes mais gerais com métodos que satisfazem esses requisitos.

(h) A ferramenta pretende fornecer assistência à fase de projeto (orientado a objetos), facilitando a identificação e reutilização de classes que possam ser incorporadas num modelo computacional de objetos [GIR89]. É de interesse conhecer as classes disponíveis para reutilização antes da fase de projeto [PRO89]. Porém, a utilização da ferramenta na fase de análise de requisitos é objetivo de pesquisa futura.

3 Esquemas de classificação

A efetividade do mecanismo de recuperação depende, principalmente, de uma boa estruturação da biblioteca de classes, ou seja, de como as classes foram classificadas e organizadas na biblioteca.

Dois mecanismos de classificação são considerados para estruturar a biblioteca de classes:

- um mecanismo de *classificação implícita*, derivado da estruturação hierarquica das classes, que permite inspecioná-las em diferentes níveis de abstração, através do mecanismo de busca exploratória;
- um mecanismo de *classificação explícita*, onde cada classe é descrita através de um conjunto de descritores mediante os quais as classes são recuperadas pelo mecanismo de recuperação sistemática.

A hierarquia de classes da biblioteca estabelece, implicitamente, um critério de classificação das classes da biblioteca. Os componentes (classes) são organizados como grafos de herança (estrutura de árvore, no caso de herança simples) fornecendo componentes em múltiplos níveis de abstração.

Os esquemas de classificação de componentes de software propostos na literatura [PRI87, FRA87, SUG86, TEI 89] fornecem critérios para a classificação de componentes pela sua funcionalidade. Esses critérios não resultam suficientes para descrever a semântica de uma classe de objetos que, além de funções, incorpora dados.

Consideramos que, basicamente, uma classe pode ser descrita através dos seguintes esquemas de classificação [EMB87]:

(1) Um esquema de classificação que descreve, através de um conjunto de palavras chaves, o *domínio de aplicação* da classe.

(2) Um esquema de classificação que descreve, através do nome da classe e de um conjunto de sinônimos e palavras chaves, as *características gerais* da classe.

(3) Um esquema de classificação que descreve, através de um

conjunto de palavras chaves, a *estrutura de dados* da classe.
(4) Um esquema de classificação que descreve a *interface da classe* (o *que* os métodos fazem), através dos nomes dos métodos, de sinônimos desses nomes e de palavras chaves que descrevem a funcionalidade de cada um dos métodos na interface.

Cada descritor, nos esquemas de classificação (1), (2), (3) ou (4) tem associado um coeficiente, denominado *coeficiente de certeza (CC)*, que indica o grau de certeza com que o descritor descreve a semântica da classe, segundo o esquema de classificação correspondente. O valor desse coeficiente é atualizado com os sucessivos usos da ferramenta e é determinado em função do número de seleções (NSs) e do número de rejeições (NRs) de uma classe, segundo esse descritor. O seguinte critério será implementado inicialmente:

(a) $CC = NS / (NS + NR) \in (0,1]$ se $NS > 0$

(b) $CC = 0$ se $NS = 0$

4. Biblioteca de classes

É definida uma estrutura de conhecimento para a biblioteca de classes que evolue através dos sucessivos usos da biblioteca, representando a informação associada à hierarquia de classes e aos diferentes esquemas de classificação definidos.

É utilizado um modelo de rede semântica [WOO75] para organizar e representar a estrutura de conhecimento da biblioteca (ver fig. 1). Os construtores básicos da rede são nodos, que descrevem os diferentes tipos de informação armazenada e conexões entre nodos, que descrevem os diferentes tipos de relacionamento entre os nodos. A vantagem de utilizar este modelo é a possibilidade de definir e atualizar dinamicamente diferentes tipos de vínculos, facilitando a extensão futura do conhecimento armazenado na biblioteca.

Definimos relações 1-n, por exemplo, as conexões *parte-de*, e 1-1, por exemplo, a relação *catalogada-como*.

O nodo *classe-de-objetos* e o relacionamento *subclasse-de* organizam a hierarquia de classes da biblioteca (observe-se que somente é considerada herança simples). O nodo *método* e a conexão *tem* representam a interface e implementação dos métodos de cada classe.

O descritor geral de uma classe, através da conexão *catalogada-como*, é representado com o nodo *descritor-classe*. A conexão *catalogado-como* representa o relacionamento entre um método e um ou mais descritores de sua funcionalidade.

Os componentes do descritor geral da classe são representados pelas conexões *parte-de* e os nodos *descritor-do-dominio-de-aplicação*, *descritor-da-classe*, *descritor-da-estrutura-de-dados* e *descritor-do-método*, associados aos esquemas de classificação *a*, *b*, *c* e *d*, respectivamente, definidos na seção 3. As conexões *parte-de* são de tipo 1-n, indicando que pode existir mais de um descritor associado a cada esquema de classificação.

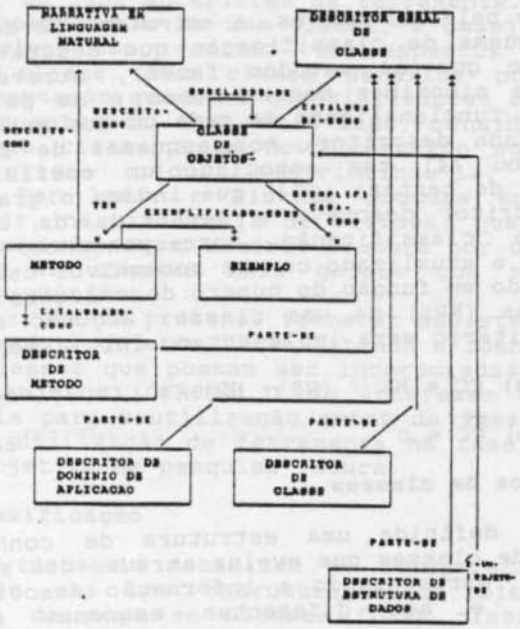

Figura 1 - Estrutura de conhecimento da biblioteca de classes

A conexão *é-um-objeto-de*, entre os nodos *descritor-da-estrutura-de-dados* e *classe-de-objetos* indica que cada descritor da estrutura de dados da classe referencia um objeto de uma outra classe.

As classes de objetos e seus métodos estão associadas com exemplos, representados na rede pelo nodo *exemplo* e as conexões *exemplificadas-como*. É sugerido que os exemplos sejam componentes de código, de pequeno tamanho, bem documentados e executáveis que ilustrem usos típicos das classes ou dos métodos.

Analogamente, classes e métodos estão associadas com descrições em linguagem natural, representados na rede pelo nodo *narrativa-em-linguagem-natural* e as conexões *descrita-como* e *descrito-como* para classes e métodos, respectivamente. A narrativa descreve classes ou métodos, sem restrições de formato ou conteúdo, e deveria ter suficiente clareza como para que métodos e classes possam ser compreendidos pelos usuários.

5. Arquitetura da ferramenta

O processo de reuso de classes é abordado através de três subprocessos: *recuperação*, *avaliação* e *aquisição de conhecimento* (ver fig. 2).

A partir de um conjunto de descritores fornecidos

pelo usuário, descrevendo a classe hipotética procurada, é iniciado o processo de recuperação (sistemática ou exploratória), utilizando o conhecimento armazenado na biblioteca de classes. O resultado deste processo é um conjunto de classes candidatas a serem reutilizadas.

As classes recuperadas pelo sistema, via recuperação sistemática ou busca exploratória, são avaliadas pelo usuário para escolher aquela que melhor se adapta a suas necessidades. O processo de avaliação é assistido com informação armazenada na biblioteca.

Como resultado do processo de avaliação pode ser selecionada uma classe e rejeitadas as demais classes candidatas. Em função da seleção ou rejeição de uma classe, o conhecimento da biblioteca é incrementado e atualizado a partir dos descritores especificados pelo usuário.

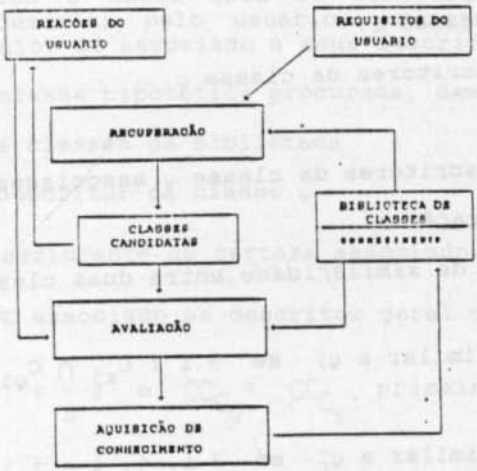

Figura 2 - Arquitetura global da ferramenta

5.1 Processo de recuperação

O processo de recuperação é suportado por dois mecanismos complementares:
- um mecanismo de *recuperação sistemática*, onde o usuário descreve a classe procurada mediante descritores, a partir dos quais é recuperado, automaticamente, um conjunto de classes candidatas, e
- um mecanismo de *busca exploratória*, onde o próprio usuário seleciona uma classe inspecionando a hierarquia de classes da biblioteca.

5.1.1 Recuperação sistemática

No mecanismo de recuperação sistemática, o usuário especifica o número máximo de classes a serem recuperadas. A ordem em que as classes recuperadas são apresentadas está baseada na similaridade dessas classes com os requisitos especificados.

Duas classes são similares se têm descritor(es) em comum em algum dos esquemas de classificação definidos. O grau de similaridade estará dado pelo número de esquemas de classificação onde se compartilham descritores. Sejam:

α - a classe hipotética descrita pelo usuário

ψ - uma classe da biblioteca

C_α - conjunto de descritores da classe α

$C_\alpha = \bigcup_i C_{\alpha i}$

$C_{\alpha i}$ - conjunto de descritores da classe α associados ao esquema de classificação i.

C_ψ - conjunto da descritores da classe ψ

$C_\psi = \bigcup_i C_{\psi i}$

$C_{\psi i}$ - conjunto de descritores da classe ψ associados ao esquema de classificação i.

A relação de *similaridade* entre duas classes é definida como:

(1) $\alpha \underset{1}{\approx} \psi$ (α é 1-similar a ψ) se $\exists\, i : C_{\alpha i} \cap C_{\psi i} \neq \emptyset$

(2) $\alpha \underset{2}{\approx} \psi$ (α é 2-similar a ψ) se $\exists\, i, j, i \neq j :$

$$C_{\alpha i} \cap C_{\psi i} \neq \emptyset \quad e$$

$$C_{\alpha j} \cap C_{\psi j} \neq \emptyset$$

(3) $\alpha \underset{3}{\approx} \psi$ (α é 3-similar a ψ) se $\exists\, i, j, k, i \neq j, i \neq k, j \neq k :$

$$C_{\alpha i} \cap C_{\psi i} \neq \emptyset \quad e$$

$$C_{xj} \cap C_{yj} \neq \emptyset \quad e$$

$$C_{xk} \cap C_{yk} \neq \emptyset$$

(4) $x \approx_4 y$ (x é 4-similar a y) se $\forall\, i\ C_{xi} \cap C_{yi} \neq \emptyset$

Informalmente, duas classes são *1-similares* se têm algum descritor em comum em pelo menos um dos esquemas de classificação definidos. São *2-similares* se têm algum descritor em comum em pelo menos dois esquemas de classificação. São *4-similares* se têm algum descritor em comum em todos os esquemas de classificação.

Os CCs associados aos descritores das classes na biblioteca, definem uma subordem entre as classes recuperadas que têm o mesmo grau de similaridade com os requisitos especificados pelo usuário. Dessa forma, se duas classes da biblioteca têm o mesmo grau de similaridade com a classe hipotética descrita pelo usuário, primeiro é recuperada a classe com maior CC associado a seus descritores. Sejam,

x - a classe hipotética procurada, descrita pelo usuário

y, z - duas classes da biblioteca

$d \in C_y$ - um descritor da classe y

CC_d - o coeficiente de certeza associado ao descritor d

CC_{C_y} - o CC associado ao descritor geral da classe y

se $x \approx_s y$ e $x \approx_s z$ e $CC_{C_y} \geq CC_{C_z}$, primeiro é recuperada a classe y.

O CC associado ao descritor geral de uma classe y é calculado em função dos CCs dos descritores comuns das classes x e y, ou seja, em função dos CCs dos descritores que foram efetivamente utilizados no processo de recuperação. O seguinte critério será adotado, inicialmente, para o cálculo do CC associado ao descritor geral de uma classe:

$$CC_{Cy} = \sum_{\forall\, d \in C_x \cap C_y} CC_d \quad / \quad \#(C_x \cap C_y)$$

com CC_d calculado em função do NSs e do NRs da classe através do descritor *d* em recuperações anteriores, segundo as fórmulas (a) e (b) na seção 3.

Por outro lado, a ferramenta recupera a classe mais

geral na hierarquia que satisfaz os requisitos do usuário. Se duas classes da biblioteca têm o mesmo grau de similaridade com a classe hipotética descrita pelo usuário; as duas classes estão relacionadas hierarquicamente e o CC associado à superclasse é maior que o CC associado à subclasse, então somente é recuperada a superclasse. Caso contrário, as duas classes são recuperadas. De uma outra forma,

se $x \approx_s y$, $x \approx_s z$, z subclasse de y e $CC_y \geq CC_z$, somente y é recuperada. Em caso contrário, ambas y e z são recuperadas.

O mecanismo de recuperação sistemática interage com o usuário através de uma interface, onde o usuário especifica o descritor geral da classe procurada, indicando os descritores associados aos diferentes esquemas de classificação.

Como os descritores dos diferentes esquemas de classificação são agregados na biblioteca num descritor geral da classe, é possível que o usuário solicite assistência para a descrição da classe procurada. Nesse caso, a ferramenta procura na estrutura de conhecimento (ver fig. 1), a partir da descrição parcial já fornecida pelo usuário, possíveis descritores que o usuário poderia utilizar para completar a descrição da classe.

As classes recuperadas são apresentadas, indicando seu grau de similaridade com os requisitos especificados. O usuário pode selecionar uma das classes recuperadas para avaliá-la ou escolher o método de busca exploratória, se o processo de recuperação sistemática falha.

5.1.2 Busca exploratória

Através do mecanismo de busca exploratória, o usuário inspeciona manualmente a estrutura de conhecimento da biblioteca. A hierarquia de classes fornece um guia para a análise das classes disponíveis, em diferentes níveis de abstração, permitindo partilhar o espaço de busca e ganhar conhecimento da informação disponível. O mecanismo permite visualizar a estrutura de dados, interface e implementação dos métodos das classes na hierarquia, assim como exemplos e descrições de classes ou métodos.

A busca exploratória é o mecanismo utilizado quando não é possível recuperar uma classe adequada através da recuperação sistemática, já porque o usuário não consegue descrever apropriadamente a classe procurada ou porque o conhecimento do sistema é insuficiente para atender os requisitos especificados. Também pode ser o mecanismo principal a utilizar-se com bibliotecas pequenas, suficientemente conhecidas pelo usuário.

5.2 Processo de avaliação

As classes candidatas obtidas através do processo de recuperação são, opcionalmente, avaliadas pelo usuário, de forma a selecionar aquela mais adequada para a aplicação em desenvolvimento.

O processo de avaliação é assistido por um "browser" de classes e pelo conhecimento existente na biblioteca,

permitindo:
(a) Examinar documentação, exemplos, estrutura de dados, bem como interface e implementação dos métodos da classe.
(b) Executar exemplos disponíveis de forma a observar o comportamento esperado de uma classe ou método.
(c) Enviar mensagens aos objetos de determinada classe e observar seu comportamento.
(d) Enviar mensagens aos objetos de uma classe criada especializando uma classe já existente.

5.3 Processo de aquisição de conhecimento

A partir do processo de avaliação das classes candidatas recuperadas, o conhecimento do sistema é incrementado e/ou atualizado em função dessa avaliação.

A aquisição de conhecimento é transparente ao usuário, ou seja, ele somente indica, no processo de avaliação, qual a classe selecionada.

Duas fontes de aquisição de conhecimento são consideradas: aquisição de conhecimento *por exemplo* e aquisição de conhecimento *por falha e sucesso* [ARA88b].

5.3.1 Aquisição de conhecimento por exemplo

A aquisição de conhecimento por exemplo ocorre quando o mecanismo de recuperação sistemática falha, porque falta conhecimento na biblioteca para selecionar classes segundo os requisitos especificados, ou quando as classes recuperadas não satisfazem ao usuário. Em consequência, o usuário pode selecionar uma classe através do mecanismo de busca exploratória.

Se os descritores especificados inicialmente pelo usuário, para o mecanismo de recuperação sistemática, refletem as características da classe selecionada por busca exploratória, esses descritores são utilizados para incrementar o conhecimento que a biblioteca possui da classe selecionada.

Nesse caso, todo novo descritor especificado pelo usuário é incorporado ao descritor geral da classe selecionada, associado ao esquema de classificação correspondente. Os CCs associados a esses descritores são inicializados, indicando que a classe foi selecionada uma vez, segundo esse descritor, e nunca rejeitada,

$$\forall d \in C_x,\ d \notin C_y : \quad C_y' = C_y \cup (d), \quad NR_d \longleftarrow 0, \quad NS_d \longleftarrow 1$$

5.3.2 Aquisição de conhecimento por falha ou sucesso

A aquisição de conhecimento por falha (classe rejeitada) ou sucesso (classe selecionada) ocorre quando uma ou mais classes são recuperadas através do mecanismo de recuperação sistemática. Como resultado do processo de avaliação, uma ou nenhuma classe do conjunto de classes candidatas é selecionada para reutilização. As outras classes candidatas são rejeitadas.

O conhecimento, na biblioteca, da classe selecionada e das classes rejeitadas é atualizado de forma a refletir a nova experiência de reuso, segundo os critérios a seguir.
(a) *Classe selecionada*: É incrementada a conta das seleções da

classe segundo determinado descritor, para todos aqueles descritores através dos quais a classe foi recuperada. Aqueles descritores especificados pelo usuário, não utilizados no processo de recuperação, por não estar incluídos no descritor geral da classe selecionada, são incorporados a esse descritor geral, associando-os ao esquema de classificação correspondente. Os CCs associados a esses descritores são inicializados indicando que a classe foi selecionada uma vez, segundo esse descritor, e nunca rejeitada,

$$\forall d \in C_x \cap C_y : \quad NS_d \longleftarrow NS_d + 1$$

$$\forall d \in C_x,\ d \notin C_y : C'_y = C_y \cup (d),\ NR_d \longleftarrow 0,\ NS_d \longleftarrow 1$$

(b) *Classes rejeitadas:* É incrementada a conta das rejeições de cada classe, segundo determinado descritor, para todos aqueles descritores através dos quais as classes foram recuperadas,

$$\forall d \in C_x \cap C_y : NR_d \longleftarrow NR_d + 1$$

6 Conclusões

Apresentou-se a especificação de uma ferramenta de apoio à reutilização de classes no desenvolvimento de software orientado a objetos. É definido um mecanismo de recuperação de classes que opera em duas modalidades complementares: uma recuperação rápida, através da especificação de descritores de classe e uma recuperação por "browsing", explorando as hierarquias de classe. São recuperadas as classes que se adaptam perfeitamente aos requisitos especificados, bem como as que apresentam algum grau de similaridade com esses requisitos. É incorporado conhecimento à biblioteca de classes de forma a refletir a experiência de reuso nos sucessivos usos da ferramenta. Esse conhecimento permite melhorar, incrementalmente, a eficiência do mecanismo de recuperação, tanto em acertos de busca, como em probabilidade de sucesso.

Um protótipo da ferramenta está sendo implementado no ambiente Smalltalk/V [DIG86], a fim de avaliar o desempenho do mecanismo de recuperação e para explorar, em geral, a reusabilidade de software.

Pretende-se estender a especificação apresentada, incorporando outras técnicas da área de recuperação de informação [SAL83], como casamento de padrões, e esquemas de classificação adicionais, como "signature".

Considerando o grau de relevância com que cada esquema descreve uma classe de objetos, está sendo analisada a possível modificação da relação de similaridade definida, atribuindo pesos distintos aos descritores segundo o esquema de classificação ao que estão associados.

Bibliografia

[ARA 88a] ARAPIS, C. & KAPPEL, G. Organizing Objects in an Object Software Base. In: TSICHRITZIS, Dennis C. ed. *Active Object Environments*, Genève, Centre Universitaire d'Informatique /Université de Genève, Jui. 1988, p.32-50.

[ARA 88b] ARANGO, Gustavo. *Design of a Software Reusability*

Subsystem in an Object-oriented Environment, Rapport Technique, Altair, 1988.

[DIG 86] DIGITALK INC. *Smalltalk/V. Tutorial and Programming Handbook*, nov. 1986.

[EMB 87] EMBLEY, D. W. & WOODFIELD, S. N. A Knowdlege Structure for Reusing Abstract Data Types. In: 9TH. INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, Monterey, Mar. 30-Apr. 2, 1987, *Proceedings*, IEEE Computer Society Press, 1987, p. 360-368.

[FRA 87] FRAKES, W. B. & NEJMEH, B. A. Software Reuse through Information Retrieval, *SIGIR Forum*, 21(1,2):30-36, Fall-Winter 1986/87.

[GIR 89] GIRARDI, Maria del Rosario. *Proposta de uma Metodologia de Desenvolvimento de Software Orientada a Objetos*, Porto Alegre, PGCC, 1989.(Monografia)

[GIR 90a] GIRARDI, Maria del Rosario. *Uma Ferramenta de Apoio à Reutilização de Software no Desenvolvimento Orientado a Objetos*. Tese de mestrado, Porto Alegre, PGCC, 1990. (Em andamento).

[GIR 90b] GIRARDI, Maria del Rosario & PRICE, Roberto Tom. O Paradigma de Desenvolvimento por Objetos, *Revista de Informatica Teórica e Aplicada*, v.1, n. 2, Porto Alegre, 1990.

[GIR 90c] GIRARDI, Maria del Rosario & PRICE, Roberto Tom. Ambientes com Suporte ao Paradigma de Desenvolvimento por Objetos. In: XIII CONGRESSO NACIONAL DE INFORMATICA (SUCESU'90), *Anais*, Rio de Janeiro, 27-31 Ago. 1990.

[MOT 86] MOTRO, A. BAROQUE: A Browser for Relational Databases, *ACM TOIS*, New York, 4(2):164-181, 1986.

[PIN 88] PINTADO, X & TSICHRITZIS, D. An Affinity Browser. In: TSICHRITZIS, Dennis C. ed. *Active Object Environments*, Genève, Centre Universitaire d' Informatique / Université de Genève, Juin 1988, p.51-60.

[PRO 89] PROFROCK, A. et alii. ITHACA: An Integrated Toolkit for Highly Advanced Computer Applications. In: OBJECT-ORIENTED DEVELOPMENT, TSICHRITZIS Dennis C. ed., Genève, Centre Universitaire d' Informatique / Université de Genève, 1989, p. 321-44.

[PRI 87] PRIETO-DIAZ, Rubén & FREEMAN, Peter. Classifying Software for Reusability. *IEEE Software*, Los Alamitos, 4(1):6-16, Jan. 1987.

[RAJ 89] RAJ, R. K. & LEVY, H. M. A Compositional Model for Software Reuse, *The Computer Journal*, 32(4):312-322, Aug. 1989.

[ROT 87] ROTENBERG, Hélio B. *Programacão Orientada a Objetos: Um Enfoque da Engenharia de Software*,Dissertação de Mestrado, PUC/RJ, 1987.

[SAL 83] SALTON, Gerard & Mc. GILL, Michael. *Introduction to Modern Information Retrieval*, Mc Graw-Hill, New York, 1983, 448 p.

[SUG 86] SUGIMOTO, H. Logic-Based Retrieval and Reuse of Software Modules. In: 5TH. ANNUAL INTERNATIONAL PHOENIX CONFERENCE ON COMPUTERS AND COMMUNICATIONS (PCCC'86). *Proceedings*. Washington, IEEE Computer Society Press, Mar. 1986.

[TAK 90] TAKAHASHI, Tadao et alii. *Programação Orientada a Objetos*, São Paulo, VII Escola de Computação, 1990.

[TEI 89] TEIXEIRA, M. & VELASCO, F. Uma Ferramenta para Auxílio na Reutilização de Software. In: III SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE, Recife, 25-27 Out. 1989, p. 253-268.

[TSI 89] TSICHRITZIS, D. Object-Oriented Development for Open Systems. In: OBJECT-ORIENTED DEVELOPMENT, TSICHRITZIS Dennis C. ed., Genève, Centre Universitaire d' Informatique / Université de Genève, 1989, p. 1-13.

[WOO 75] WOODS, W. A. What's in a link: Foundations of semantic Networks. In: Representation and Understanding. D.G. Bobrow and A. Collins eds., Academic Press, 1975, p. 35-82.

[WOO 88] WOOD, M. & SOMMERVILLE, I. A Knowdlege-Based Software Components Catalogue. In: THIRD ANNUAL CONFERENCE ON SOFTWARE ENGINEERING ENVIRONMENTS, Staffordshire, Apr. 8-10, 1987, *Proceedings*. University of Keele, Pearl Brereton ed., 1988, p. 116-131.