UM AMBIENTE PARA SONORIZAÇÃO NÃO INTRUSIVA DE APLICAÇÕES ORIENTADAS A OBJETOS

ANDRÉ LUIZ COSTA BALLISTA
ROBERTO TOM PRICE

Instituto de Informática - CPGCC
Universidade Federal do Rio Grande do Sul
Campus do Vale - Bloco IV
Av. Bento Gonçalves, 15064, CEP 91501 - 970
Porto Alegre - Rio Grande do Sul
fone: +55 (051) 336-8399 - Ramal 6161
e-mail: BALLISTA@INF.UFRGS.BR e TOMPRICE@INF.UFRGS.BR

ABSTRACT

***Marola**, a framework for the sonification of object oriented applications, is briefly described. Besides the generalised class structure for the creation of auditory interfaces **Marola** also offers a toolkit for the interactive creation of sonic classes at all levels of the framework. Auditory interfaces developed with this framework may be attached to Smalltalk applications through a reflective computational mechanism, without the need of instrumenting the application classes with any sort of code.*

***key-words:** object oriented programming, computational reflection, auditory interfaces, framework, toolkit*

1. INTRODUÇÃO

Atualmente a computação interativa baseia-se principalmente na comunicação visual. A apresentação gráfica de informações possibilita aos usuários visualizarem e manipularem o resultado de suas atividades de forma mais clara e direta [19]. O uso exclusivo do canal visual, entretanto, pode introduzir falhas na interação do usuário, gerando situações onde informações são perdidas ou confundidas devido a dificuldade de localização de suas representações visuais [18]. Interfaces ampliadas com som podem ser utilizadas para distribuir melhor as informações entre os diferentes canais sensoriais, ampliando com isto a eficácia da comunicação homem-máquina. Interfaces sonoras podem envolver, por exemplo, o uso de síntese e reconhecimento de voz, geração musical, trilhas sonoras, ou efeitos sonoros na forma de ruídos e sinais.

A sonorização[1] de interfaces é uma atividade limitada por muitos problemas. Entre os principais estão (a) a falta de padrões para a especificação de *hardware* ou *software* de síntese de som, tornando difícil a definição de sonoridades[2] que mantenham suas características através do grande número de plataformas computacionais existentes [7]; (b) a ausência de modelos computacionais capazes de representar e estruturar as sonoridades, obrigando que a maioria das experiências práticas de sonorização sejam produzidas de forma *ad-hoc* [15]; (c) a falta de regras que direcionem o processo de adição do som em interfaces, comprometendo com isto a independência entre os objetos de diálogo e os objetos da aplicação; e (d) a falta de ferramentas apropriadas para o desenvolvimento de aplicações sonorizadas.

Neste trabalho é apresentado **Marola**, um *framework* [21,23] para a sonorização de aplicações orientadas a objetos desenvolvido na plataforma Smalltalk Windows. Interfaces sonoras desenvolvidas com este *framework* podem ser associadas a aplicações Smalltalk através de um mecanismo de reflexão computacional [14,25] específico. Desta forma, mesmo aplicações já existentes podem ser sonorizadas sem a necessidade de instrumentação[3] de seus códigos fonte. Além de uma hierarquia de classes, **Marola** fornece também um *toolkit* [21] para definição interativa de novas classes e para o desenvolvimento de aplicações sonorizadas. Através deste *toolkit* novas classes podem ser interativamente adicionadas ao *framework* sem a necessidade de manipulação extensiva de código fonte. Um conjunto de classes básicas, diretamente conectadas às funções multimídia do ambiente Windows, fornecem acesso aos diferentes geradores de som para as classes do *framework*.

O restante deste trabalho está organizado como segue. Na seção 2 é apresentada a arquitetura geral do *framework*. Na sequência é sucintamente descrito o comportamento do mecanismo de reflexão computacional implementado. Na seção 4 um exemplo da sonorização é desenvolvido, e na seção 5 são apresentadas algumas considerações sobre o uso do ambiente, possíveis extensões e, particularmente, sua aplicabilidade em Engenharia de *Software*.

---

[1] O termo sonorizar se refere ao processo de adição do som a aplicações computacionais.

[2] O termo sonoridade é utilizado para referenciar características de diferentes sensações auditivas.

[3] Usualmente instrumentar uma aplicação computacional equivale a adicionar anotações, textuais ou não, em seu código fonte com o objetivo de auxiliar no processo de depuração. Neste trabalho o termo significa modificar o código fonte de uma aplicação afim de incluir novas funcionalidades não previstas originalmente.

2. A Arquitetura Geral do Framework Marola

As classes definidas pelo Marola são organizadas em uma arquitetura dividida em quatro camadas. Na figura 1 são representadas as principais classes de cada camada.

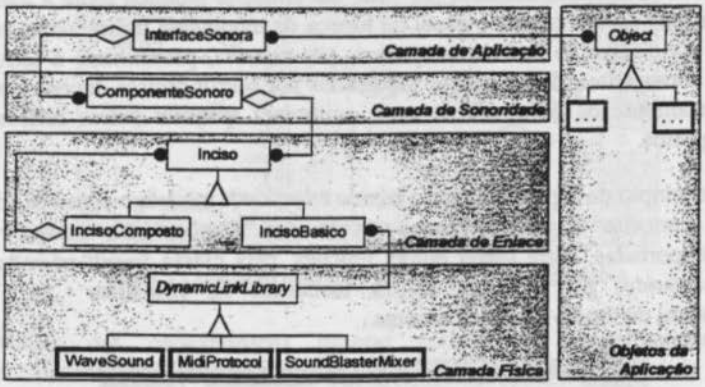

figura 1 - Arquitetura do *framework* Marola

Na notação utilizada [17] a representação de classes é acrescida dos seguintes grafismos: retângulos simples indicam classes abstratas definidas pelo *framework*; retângulos largos indicam classes concretas definidas pelo *framework*; retângulos duplos indicam classes concretas definidas pelo usuário; nomes em itálico indicam classes pré-definidas do ambiente Smalltalk.

2.1 A Camada Física

As classes da Camada Física modelam o acesso a cada um dos equipamentos de geração de som disponíveis para uso. Detalhes de manipulação destes equipamentos são encapsulados nos métodos de controle e monitoramento exportados pelas classes desta camada. Nesta versão do Marola o acesso aos equipamentos de geração de som foi implementado através de bibliotecas de linkagem dinâmica (DLL). No ambiente Smalltalk, por sua vez, cada DLL é modelada sob a forma de uma subclasse da classe pré-definida *DynamicLinkLibrary*. As classes implementadas possibilitam o acesso a sintetizadores e digitalizadores de som, equipamentos MIDI e unidades de audio-disco digital.

2.2 A Camada de Enlace

Na Camada de Enlace são modeladas classes de incisos. Incisos são estruturas computacionais abstratas que possibilitam o encapsulamento de coleções de entidades discretas em um único elemento. A estruturação do som em incisos permite descrever uniformemente, e em qualquer nível de detalhamento, desde uma nota específica até efeitos sonoros especiais

como um *fade-in*[4]. Dois tipos de incisos podem ser construídos no *framework* **Marcola**: incisos básicos e incisos compostos.

Cada classe de inciso básico está associada à uma única classe da Camada Física, e modela, através dos serviços oferecidos por esta, um conceito atômico sobre a manipulação do som. Incisos básicos desempenham o papel de blocos de montagem utilizados na construção de outros objetos. As classes de incisos compostos, por sua vez, possibilitam a estruturação, em conceitos mais abrangentes, dos conceitos modelados por outros incisos. Incisos compostos não manipulam diretamente os geradores de som, controlam somente outros incisos sejam eles básicos ou compostos.

No exemplo da figura 2, o inciso básico *Intensidade* modela o conceito de intensidade do som. Este conceito é implementado através das funções *seta volumeDireito* e *seta volumeEsquerdo* exportadas, entre várias outras funções, pela classe *SoundBlasterMixer*. O inciso composto *Deslocamento*, por sua vez, utiliza instâncias de *Intensidade* para modelar o deslocamento lateral dinâmico de fontes sonoras.

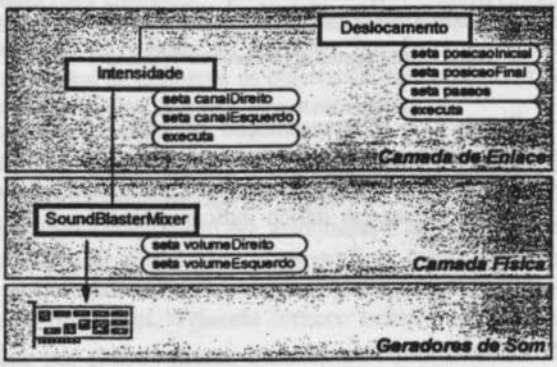

figura 2 - Exemplo da modelagem de incisos

2.3 A Camada de Sonoridade

Na Camada de Sonoridade são modelados e implementados os componentes sonoros de interface. Usualmente, em interfaces gráficas, cada componente de interface pode ser individualmente configurado, visando sua adequação à interface na qual será utilizado. De forma análoga, componentes sonoros também podem ser individualmente configurados. Esta configuração pode envolver, por exemplo, a definição de intensidade, timbre, ritmo, duração ou localização espacial de determinados efeitos.

Cada componente sonoro representa a abstração de possíveis características sonoras existentes em diferentes aplicações. A figura 3 ilustra este processo de abstração. No diagrama 1, diferentes aplicações, representadas por retângulos, apresentam conjuntos de características sonoras próprias, representadas por balões. Quando confrontados, estes conjuntos de

[4] Característica de uma fonte sonora que é gradualmente percebida através da variação de sua intensidade.

características podem revelar, como representado no diagrama 2, certas similariedades. Componentes sonoros são definidos a partir destas similariedades, possibilitando com isto a criação de bibliotecas de padrões de características sonoras. Estas bibliotecas, por sua vez, podem ser utilizadas no processo de sonorização de novas aplicações.

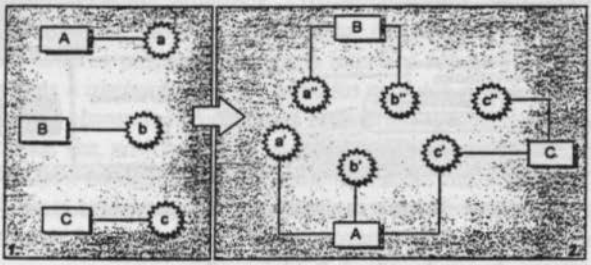

figura 3 - Abstração do comportamento sonoro das aplicações

Muitas das caracteritiscas sonoras apresentam comportamentos dinâmicos. Componentes sonoros modelam características sonoras dinâmicas através de Diagramas de Estados [12,16,20,24]. Com o uso de componentes sonoros, o comportamento sonoro de diferentes aplicações pode ser definido pelo agrupamento dos comportamentos individuais de cada componente sonoro utilizado. Diagramas de Estados utilizam uma notação mais concisa do que a utilizada por diagramas de transição de estado, suportando a decomposição hierárquica, a definição incremental, o controle de fluxo e concorrência explícita [12]. Por isto Diagramas de Estados parecem ser alternativas mais atraentes para a modelagem de sistemas reativos, como a interação homem-máquina por exemplo.

Cada componente sonoro é formado por um conjunto de incisos da Camada de Enlace, e por um Diagrama de Estados, que controla sua dinâmica. Ações de transições e atividades de estados, descritas como métodos do próprio componente sonoro, são automaticamente ativadas durante a evolução do componente por seus estados e transições. Na implementação destes métodos, mensagens podem ser enviadas aos incisos utilizados, provocando por exemplo, a ativação, desativação e modificação das propriedades destes incisos e consequente variação da sonoridade percebida.

No diagrama de estados da classe `DisparadorSonico`, ilustrado na figura 4, a mensagem *dispara* está associada a transição *clic*, e será enviada para o próprio componente sempre que esta vier a ocorrer. Através do método *eventoxargumento:* o diagrama de estado dos componentes sonoros é alimentado com eventos. Este método é normalmente ativado pelas classes da Camada de Aplicação após a interceptação de métodos refletidos.

Um conjunto de eventos pode ser definido para cada classe de componente sonoro. Outras classes que utilizem estes componentes sonoros podem, por sua vez, associar mensagens a cada um destes eventos, possibilitando que métodos específicos sejam evocados na ocorrência destes eventos. Através deste mecanismo simples é possível que outras computações, não necessariamente relacionadas com a produção ou controle de som, sejam ativadas na aplicação sonorizada.

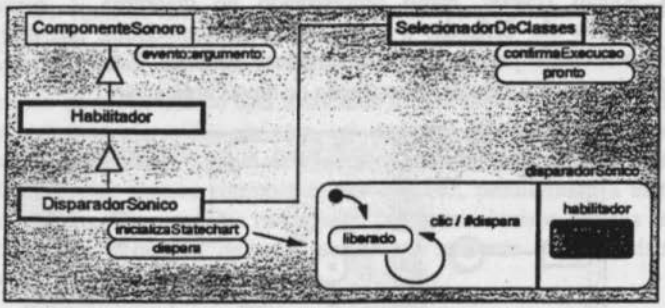

figura 4 - Exemplo da modelagem de componentes sonoros

Os diagramas de estados são definidos graficamente e automaticamente convertidos, pela ferramenta de desenho do ambiente **Marola**, no método **inicializaStatechart** de cada classe de componente sonoro. Um interpretador de diagramas de estados, especialmente implementado para uso neste *framework*, é responsável pela evolução dos diagramas de cada instância de componente sonoro utilizada. O comportamento modelado pelos diagramas de estados é herdado através da hierarquia de componentes sonoros.

2.4 Camada de Aplicação

Na Camada de Aplicação são implementadas classes de interfaces sonoras. Estas interfaces sonoras podem ser associadas, através de um mecanismo de reflexão computacional [14,25] específico, a aplicações desenvolvidas no ambiente Smalltalk. O uso do mecanismo de reflexão elimina a necessidade de instrumentação de aplicações para uso dos aspectos sonoros, possibilitando também a sonorização de aplicações já existentes. Instâncias das subclasses de *InterfaceSonora* atuam como meta-objetos, monitorando a troca de mensagens entre as classes, interceptando métodos específicos e ativando a execução de métodos refletidos.

Interfaces Sonoras são formadas por um conjunto de componentes sonoros, selecionados dentre as classes da Camada de Sonoridade, e uma lista de reflexão. Nesta lista são enumerados os métodos da aplicação, juntamente com suas classes, que devem ser interceptados pelo mecanismo de reflexão. No instante em que um destes métodos é ativado o mecanismo de reflexão, descrito na seção 3, transfere o controle para as interfaces sonoras que realizam o processamento das características sonoras da aplicação.

2.5 Objetos Da Aplicação

Entre os objetos da aplicação podem estar as classes utilizadas no desenvolvimento de interfaces gráficas, e as classes que representam o domínio da aplicação. Através do mecanismo de reflexão, descrito a seguir, qualquer tipo de classe pode ser monitorada, possibilitando o desenvolvimento de interfaces sonoras dissociadas de outros tipos de interface.

3. O Mecanismo de Reflexão do Framework Marola

O mecanismo de reflexão computacional implementado, utiliza uma adaptação da técnica de interceptação de mensagens [1,5,14] para a modificação dos dicionários de métodos associados às classes do ambiente Smalltalk. O funcionamento deste mecanismo é ilustrado através da figura 5. Inicialmente são definidas as listas de métodos a serem interceptados, com base nos componentes sonoros utilizados por cada interface sonora (1). Quando ativadas, cada interface sonora programa a interceptação dos métodos contidos na sua lista (2). No instante em que métodos refletidos são chamados o controle é transferido automaticamente para as interfaces sonoras (3). Estas, por sua vez, notificam seus componentes sonoros do ocorrido (4). O diagrama de estados do componente é modificado produzindo alterações nas sensações sonoras ativas (5). Após o processamento de todas as interfaces sonoras o controle é retornado à classe interceptada, o método original é executado e o fluxo normal de processamento restaurado (6).

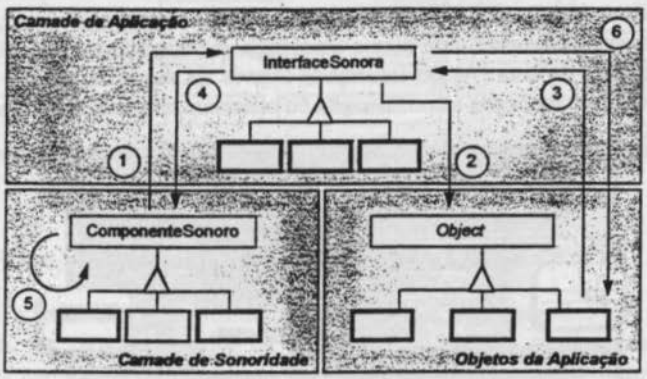

figura 5 - Funcionamento do mecanismo de reflexão

4. Construindo Aplicações Sonorizadas

No desenvolvimento de aplicações sonorizadas duas grandes etapas estão envolvidas: a concepção e a implementação da interface sonora. Na primeira etapa os estudos de acústica [26], de psicoacústica [28], e o relato de experiências prévias [3,4,6,8,9,10,11,13], podem orientar na escolha do estilo de sonorização e das sonoridades a serem utilizadas. Na segunda etapa ferramentas específicas, incluindo o próprio Marola, podem auxiliar na modelagem e implementação das sonorizações das aplicações.

Os passos básicos para o desenvolvimento de interfaces sonorizadas, através do *framework* Marola, podem ser resumidamente descritos da seguinte forma:

A) Projetar o estilo da sonorização. Nesta etapa é observado o funcionamento geral da aplicação em busca de características e comportamentos específicos a serem sonorizados.

B) Projetar as diferentes sonoridades. Implementar componentes sonoros que representem os comportamentos e características observados. Estes componentes podem ser projetados como extensões de componentes já existentes entre as classes do *framework*.

C) Agregar em uma interface sonora os diferentes componentes sonoros que representem o comportamento sonoro da aplicação.

D) Configurar o mecanismo de reflexão computacional. Nesta configuração são definidos os métodos específicos da aplicação a serem interceptados pelo mecaniso de reflexão.

4.1 Sonorizando o Selecionador de Classes

Na figura 6 é apresentada a interface gráfica do Selecionador de Classes, uma aplicação orientada a objetos, mono-classe, desenvolvida no ambiente Smalltalk, para possibilitar ao usuário selecionar visualmente uma classe específica. A classe desejada pode ser selecionada interativamente através da lista (A) ou informada através da digitação de seu nome no campo (B). Com os botões (C) é realizada a confirmação ou cancelamento da seleção realizada.

figura 6 - Aplicação exemplo

Os balões da figura 6 representam uma sonorização adicionada à interface. Sinais sonoros distintos, CLUNK e CLUNK, são associados a cada um dos botões (C). Uma nota musical é associada à lista de seleção (A). Esta nota é emitida sempre que uma linha é selecionada. A tonalidade desta nota dependende da linha selecionada, assim, a primeira linha está associada ao dó, a segunda ao dó sustenido, e assim sucessivamente. Na figura 7 estão ilustradas as classes definidas para esta sonorização, suas associações e, através de arcos direcionados, suas colaborações.

O `DisparadorSonico` utiliza o inciso básico `SomDigital` para reproduzir arquivos de som digitalizados. Dois disparadores, configurados para reproduzir diferentes arquivos, são utilizados na interface sonora. O `ContadorSonico`, por sua vez, está associado ao inciso composto `NotaMusical` que utiliza os incisos básicos `NoteOn` e `NoteOff` para ativar diferentes notas nos sintetizadores MIDI [27]. Os métodos *ok* e *cancela* da aplicação são interceptados, pela interface sonora `SelecionadorDeClassesSonicos`, através do mecanismo de reflexão. Quando ativados, o controle é

transferido para a interface sonora através do método *processaMensagem*. A interface notifica então, através do método *evento:argumento:*, os componentes pertinentes. Estes, devido a modificação de seus estados, ativam os incisos através dos métodos *executa* ou *executaAgora*, promovendo variações na sensação sonora percebida.

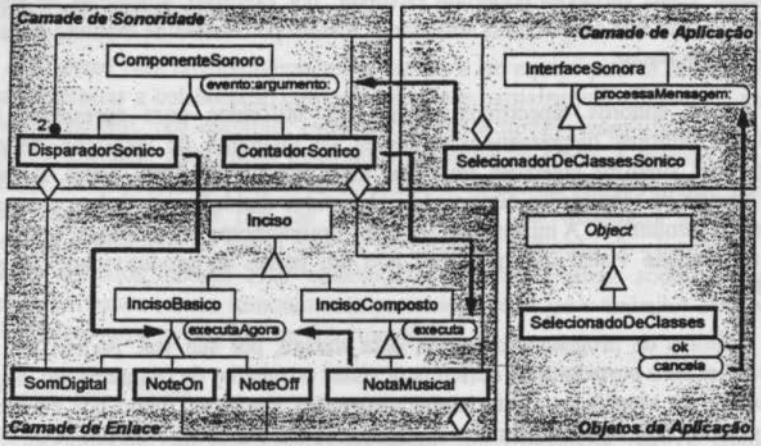

figura 7 - Classes obtidas no desenvolvimento da sonorização

5. Possíveis Usos do Som na Engenharia de Software

Com a expansão da multimídia, o uso do som nos computadores passou a ser uma realidade prática. Os rápidos avanços da tecnologia de geração e manipulação do som tem tornado cada vez mais viável, e atrativo, o desenvolvimento de aplicações ampliadas com som. Apesar disto, apenas uma pequena fração das aplicações computacionais existentes utilizam o canal auditivo para apresentar informações a seus usuários. Na engenharia de *software* é possível imaginar o uso do som em diferentes áreas.

5.1 Sonorizando Ferramentas de Desenvolvimento

Durante o desenvolvimento de aplicações é comum serem acrescentadas anotações, textuais ou sonoras, ao código fonte dos programas com o objetivo de facilitar a depuraração dos mesmos [18]. Muitas vezes, no entanto, estas anotações acabam inadvertidamente introduzindo novas fontes de erros. Ferramentas de depuração aplicam a noção de *breakpoints* como forma segura de realizar anotações sobre programas. *Breakpoints*, entretanto, podem dificultar a localização de erros, pois somente indicam, sem a análise conjunta de outras informações, pontos nos programas em que a execução é interrompida. Através do mecanismo de reflexão incorporado ao **Marola** é possível adicionar pontos de controle e inferência, sobre trocas de mensagens e acesso a propriedades por exemplo, sem a necessidade de instrumentar códigos fonte. As informações trocadas entre os objetos da aplicação podem, por sua vez, ser utilizadas na caracterização dos diferentes sinais sonoros emitidos.

O acompanhamento do desempenho de aplicações pode, por exemplo, ser realizado através de sinais sonoros específicos. Uma ferramenta para este fim poderia associar timbres e localizações espaciais específicas à cada uma das diferentes classes monitoradas. Variações na sonoridade destes timbres poderiam ser induzidas conforme o número de mensagens solicitadas de cada classe. Estas variações poderiam envolver, por exemplo, alterações de intensidade do timbre, fazendo com que classes mais solicitadas fossem percebidas de maneira mais presente do que as menos solicitadas.

Sinais sonoros específicos podem ser utilizados para apresentar informações semânticas em editores diagramáticos. Editores de diagrama de transição de estados, por exemplo, podem sinalizar conexões inválidas entre objetos gráficos, como a conexão entre transições, através de tonalidades dissonantes. Diferentes dissonâncias podem estar associadas a diferentes erros semânticos. A informação sonora é normalmente percebida de forma mais rápida do que a informação visual trazendo com isto um ganho de produtividade no processo de diagramação.

Editores de programas dirigidos pela sintaxe, por sua vez, podem ser sonorizados tornando factível para portadores de deficiência visual o trabalho de programação. Sinais sonoros distintos podem ser associados a diferentes construções sintáticas, como por exemplo laços de repetição, declarações de procedimentos, chamadas de funções e atribuições. Estes sinais poderiam ser disparados a medida que um cursor, sensível ao contexto, fosse movimentado pelo código fonte do programa.

5.2 Prototipando Interfaces

O desenvolvimento da interface gráfica das aplicações consome, aproximadamente, 70% do esforço de desenvolvimento [22]. O desenvolvimento de interfaces alternativas, entre elas as sonorizadas, pode aumentar ainda mais este percentual. Com **Mavola** a interface sonora pode ser ortogonalmente desenvolvida com a aplicação ou com outras interfaces da aplicação, aumentando desta forma a produtividade do desenvolvimento. Além disto, bibliotecas de componentes sonoros podem ser desenvolvidas facilitando e acelerando o desenvolvimento de interfaces sonoras.

Teste de interfaces sonoras são normalmente restritos a aplicações simples utilizadas por uma pequena comunidade de usuários. Através de **Mavola** interfaces sonoras mais aprimoradas podem ser facilmente criadas, possibilitando que experiências de sonorização sejam testadas em uma comunidade de usuários mais abrangente.

5.3 Substituindo Interfaces já Existentes

Aplicações embarcadas, como as existentes em carros, eletrodomésticos, aparelhos eletrônicos, ou máquinas industriais por exemplo, possuem interfaces cada vez mais complexas. Muitas destas interfaces exigem do usuário a concentração visual sobre painéis ou botões de difícil visualização e interação. Em certas situações esta exigência pode aumentar o risco de ocorrência de acidentes pessoais e danos materiais. Sinais ou comandos sonoros podem ser utilizados visando a melhoria da interação homem-máquina ou a substituição de interfaces

inadequadas à atividade envolvida. Através do Marola é possível projetar interfaces sonoras, dissociadas de quaisquer outros tipos de interfaces, e avaliar previamente a eficácia destas, facilitando o desenvolvimento de interfaces adequadas à atividade envolvida.

5.4 Facilitando o Desenvolvimento em Equipe

Ambientes para desenvolvimento de *software* em equipe podem utilizar sonorizações como forma de ampliar a colaboração entre os diferentes participantes. Classes da aplicação em desenvolvimento podem, por exemplo, estar associadas a timbres específicos que apresentem variações, de intensidade, localização espacial, tonalidade ou ritmo, a medida que alterações sejam efetuadas. Cada projetista pode estar associado a diferentes sinais sonoros, possibilitando que alterações de código fonte por ele efetuadas levem sua assinatura sônica. Atividades realizadas, como a definição de novas classes ou métodos, remoção de classes ou métodos, e modificação de métodos, podem estar associadas a diferentes sinais sonoros. Desta forma, modificações importantes seriam remotamente percebidas sem que o trabalho desenvolvido por outros membros da equipe tivesse que ser interrompido.

6. Considerações Finais

Neste trabalho foi apresentado Marola, um *framework* para a sonorização de aplicações orientadas a objetos. Aplicações desenvolvidas no ambiente Smalltalk podem ser conectadas à interfaces sonoras, desenvolvidas com o auxílio deste *framework*, através de um mecanismo de reflexão computacional específico. Desta forma, mesmo aplicações já existentes podem ser sonorizadas sem a necessidade de modificação de seus códigos fonte. Através de alguns exemplos de sonorizações, foi salientada a relevância do uso do som em ambientes de desenvolvimento de *software*.

A hierarquia de classes presentes neste *framework* possibilita a implementação de novas interfaces sonoras de forma rápida e simples. De forma geral, a implementação de novas interfaces é basicamente realizada através da seleção de componentes sonoros já existentes, e da definição de métodos da aplicação a serem interceptados pelo mecanismo de reflexão. Caso necessário, novos incisos e componentes sonoros pode ser rapidamente derivados das classes abstratas já presentes na hierarquia de classes do *framework*. Na maioria dos casos, o volume de código necessário para a implementação de métodos específicos de novas classes é bastante reduzido. Devido a definição do protocolo de comunicação entre as diferentes classes do *framework*, a tarefa de implementação de novas classes é grandemente simplificada, tornando possível inclusive, a definição de ferramentas que realizem esta codificação de forma automática ou semi-automática.

Este *framework* é acompanhado de *toolkit* próprio, denominado **TK-Marola**, que possibilita a sonorização de aplicações de forma interativa e sem a necessidade de manipulação extensiva de código fonte. O desenvolvimento de novos componentes sonoros, a definição de interfaces sonoras, e a programação do mecanismo de reflexão computacional podem ser realizadas interativamente, sendo suas implementações automaticamente produzidas, em código Smalltalk, por ferramentas específicas disponíveis neste *toolkit*.

O mecanismo de reflexão computacional implementado para o *framework* Marola possibilita que interceptações de métodos sejam dinamicamente definidas e eliminadas, sem afetar, com isto, o funcionamento normal das classes interceptadas. Desta forma, é eliminada a possibilidade de introdução de erros nas aplicações durante a busca dos métodos a serem interceptados. Esta flexibilidade e segurança, oferecidas pelo mecanismo de reflexão computacional implementado, facilitam o desenvolvimento ortogonal, entre interfaces sonoras e aplicações específicas, e valorizam o estilo incremental na prototipação de novas interfaces sonoras. No entanto, para que interfaces sonoras desenvolvidas com este *framework* sejam adequadamente associadas à aplicações específicas, é necessário um conhecimento bastante aprofundado sobre as classes e métodos destas aplicações. A utilização de uma ferramenta para documentação de *frameworks*, como a apresentada em [2], pode se revelar importante para que este conhecimento seja adquirido mais rapidamente.

Nesta versão do Marola, o comportamento sonoro das aplicações é modelado através de Diagramas de Estado. A grande vantagem desta utilização está na possibilidade de definição, através das classes de componentes sonoros, de padrões de comportamento. Quanto maior o número de componentes sonoros disponíveis no *framework*, maior o número de padrões modelados, bem como mais rapidamente novas sonorizações poderão ser implementadas e mais abrangente poderão ser estas sonorizações. A distribuição destes componentes entre pesquisadores da área de sonorização de interfaces, pode possibilitar a validação de propostas de sonorizações por uma grande comunidade de usuários.

A implementação das classes do *framework* pode ser facilmente portada para outras versões do ambiente Smalltalk, ou até mesmo para outras linguagens de programação. A portabilidade do mecanismo de reflexão, entretanto, é determinada pela existência de facilidades reflexivas na linguagem a ser utilizada. Caso estas facilidades existam, pode se tornar viável a reprodução do comportamento do mecanismo de reflexão descrito neste trabalho. A implementação das classes da camada física é específica para a plataforma de desenvolvimento utilizada. Estas devem, portanto, ser reescritas para o código nativo da plataforma a ser utilizada.

7. Referências Bibliográficas

[1] CAMPO, MARCELO RICARDO, PRICE, ROBERTO TOM. ***Meta-Objects Manager: A framework for customizable meta-object support for Smalltalk 80.*** submetido ao 1° Simpósio Brasileiro de Linguagens de Programação. Disponível através de ftp://caracol.inf.ufrgs.br/pub/amadeus/

[2] CAMPO, MARCELO RICARDO, PRICE, ROBERTO TOM. ***A Visual Reflective Tool for Framework Understanding***. Technology of Object-Oriented Languages and Systems: 19. Fevereiro 1996

[3] BREWSTER, STEPHEN A., WRIGHT, PETER C., DIX, ALAN J., EDWARDS, ALISTAIR D.N. ***The Sonic Enhancement of Graphical Buttons***. Proceedings of INTERACT'95 - 5th International Conference on Human-Computer Interaction.. 1995.

[4] BREWSTER, STEPHEN A. ***The Development of a Sonically-Enhanced Widget Set***. Proceedings EWHCI'95. 1995

[5] CAMPO, MARCELO RICARDO, PRICE, ROBERTO TOM. ***Meta-Object Support for Framework***

Understanding Tools. Workshop on Advances in Metaobject Protocols and Reflection. META'95. Realizado durante Ninth European Conference on Object-Oriented Programming, ECOOP 95. 1995

[6] LEIMANN, ERIC, SCHULZE, HANS-HENNING. ***Earcons and Icons: An Experimental Study***. Proceedings of INTERACT'95 - 5th International Conference on Human-Computer Interaction.. 1995.

[7] MADHYASTHA, TARA M., REED, DANIEL A.***Data Sonification: Do You See What I Hear?***. IEEE Software. Março 1995.

[8] RESNICK, PAUL. VIRZI, ROBERT A. ***Relief from the Audio Interface Blues: Expanding the Spectrum of Menu, List, and Form Styles***. ACM Transactions on Computer-Human Interaction. vol. 2. no. 2. Junho 1995.

[9] BREWSTER, STEPHEN A., WRIGHT, PETER C., EDWARDS, ALISTAIR D.N. ***The Design and Evaluation of an Auditory-Enhanced ScrollBar***. Proceedings of CHI'94. 1994

[10] DIX, ALAN J., BREWSTER, STEPHEN A. ***Causing Trouble with Buttons***. Ancillary proceedigns of HCI'94. 1994.

[11] KARSENTY, ALAIN. ***Interfaces Sonores pour Collecticiels***. Actes des Sixièmes Journées sur L'Ingénierie des Interfaces Homme-Machine. 1994.

[12] LUCENA, FÁBIO N., LIESENBERG, HANS K.E. ***A Statechart Engine to Support Implementations of Complex Behavior***. Anais do XXI Seminário Integrado de Software e Hardware. SEMISH'94. 1994

[13] BREWSTER, STEPHEN A., WRIGHT, PETER C., EDWARDS, ALISTAIR, D.N. ***An Evaluation of Earcons for Use in Auditory Human-Computer Interfaces***. Proceedings INTERCHI'93. Abril 1993.

[14] LISBOA, PAULO H.C., TEPEDINO, JOSÉ FERNANDO, MEIRA, SILVIO LEMOS. ***Reflexão Computacional em Smalltalk***. Anais do XX Seminário Integrado de Software e Hardware. SEMISH'93. 1993.

[15] BREWSTER, STEPHEN A. ***Providing a Model for the Use of Sound in User Interfaces***. Technical Report No. YCS 169. University of York. Department of Computer Science

[16] CANGUSSU, JOÃO W.L., MASIERO, PAULO CESAR, MALDONADO, JOSE CARLOS. ***Execução Programada de Statecharts***. Anais do VII Simpósio Brasileiro de Engenharia de Software. 1992.

[17] RUMBAUGH, JAMES, BLAHA, MICHAEL, PREMERLANI, WILLIAM, EDDY, FREDERICK, LORENSEN, WILLIAM. ***Object-Oriented Modeling and Design***. Englewood Cliffs. Prentice-Hall. 1991.

[18] MOUNTFORD, S.JOY, GAVER, WILLIAM W. ***Talking and Listening to Computers***. in The Art of Computer Human Interface Design. Addison-Wesley Publishing Company, Inc. 1990.

[19] NORMAN, DONALD A. ***Why Interfaces Don't Work***. in The Art of Computer Human Interface Design. Addison-Wesley Publishing Company, Inc. 1990.

[20] HARTSON, H. REX, HIX, DEBORAH. ***Human-Computer Interface Development: Concepts and Systems for its Management***. ACM Computing Surveys, 21 (1): 5-92. Março 1989.

[21] JOHNSON, RALPH E., FOOTE, BRIAN. ***Designing Reusable Classes***. Journal of Object-Oriented Programming. vol. 1. no. 2. Junho 1988.

[22] MEYER, BERTRAND. ***Object-Oriented Software Construction***. Prentice Hall International Series in Computer Science. 1988.

[23] DEUTSCH, L. PETER. ***Design Reuse and Frameworks in the Smalltalk-80 System***. in Software Reusability. Volume II: Applications and Experience. Edited by Ted J. Biggerstaff and Alan. J. Partis. Addison-Wesley Publishing Company. 1987

[24] HAREL, DAVID. ***Statecharts: A Visual Formalism for Complex Systems***. Science of Computer Programming. vol. 8. no. 3. Junho 1987.

[25] MAES, PATTIE. ***Concepts and Experiments in Computational Reflection***. Proceedings of the OOPSLA'87. Conference on Object-Oriented Programming Systems, Languages and Applications. 1987

[26] DODGE, CHARLES, JERSE, THOMAS A. ***Computer Music. Synthesis, Composition, and Performance***. Schirmer Books. 1985

[27] LOY, GARETH. *Musicians Make a Standard: The MIDI Phenomenon.* Computer Music Journal. vol. 9. no. 4. Winter 1985.

[28] LINDSAY, PETER H., NORMAN, DONALD A. ***An Introduction to Psychology***. Second Edition. Hearcourt Brace Jovanovich, Publishers. 1977.