UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

FERRAMENTA PARA GERAÇÃO DE PARTITURAS NO SISTEMA DE MUSICOGRAFIA BRAILE

MARCOS FERNANDO SIMON

BLUMENAU
2012

2012/2-20

MARCOS FERNANDO SIMON

FERRAMENTA PARA GERAÇÃO DE PARTITURAS NO SISTEMA DE MUSICOGRAFIA BRAILE

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciência da Computação — Bacharelado.

Prof. Aurélio Faustino Hoppe, Mestre - Orientador

BLUMENAU
2012

2012/2-20

FERRAMENTA PARA GERAÇÃO DE PARTITURAS NO SISTEMA DE MUSICOGRAFIA BRAILE

Por

MARCOS FERNANDO SIMON

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Aurélio Faustino Hoppe, Mestre – Orientador, FURB
Membro:	Prof. Marcel Hugo, Mestre – FURB
Membro:	Prof. Roberto Heinzle, Doutor – FURB

Blumenau, 10 de dezembro de 2012.

Dedico este trabalho a minha mãe Ilda Maiola Simon e ao meu pai, Hélcio Pedro Simon.

AGRADECIMENTOS

A Deus, pela proteção, seu imenso amor e graça.

A meus pais, por terem me incentivado a estudar desde pequeno.

Ao professor, Aurélio Faustino Hoppe, pela orientação, incentivo, dedicação e acima de tudo por ter acreditado no potencial de desenvolvimento deste trabalho.

Aos meus amigos, pela motivação, empurrões e cobranças.

A música exprime a mais alta filosofia numa linguagem que a razão não compreende.

Arthur Schopenhauer

RESUMO

Este trabalho apresenta uma ferramenta geradora de partituras para o sistema de musicografia braile, utilizando arquivos MIDI. No desenvolvimento da ferramenta foi utilizada a biblioteca Java Sound para efetuar a leitura e conversão dos arquivos MIDI para o sistema de musicografia braile. Os resultados obtidos demonstram a viabilidade de uma ferramenta para a geração de partituras braile.

Palavras-chave: Musicografia braile. Arquivos MIDI

ABSTRACT

This work presents a tool system for sheet music generation in braille, using MIDI files as input information. The use of Java Sound library was necessary for reading and interpreting the MIDI files, and consequently, converting to musical braille. The results demonstrate the viability of a tool for braille sheet music generation.

Keywords: Braille music. MIDI files.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

API – *Application Programming Interface*

IDE – *Integrated Development Environment*

MIDI – *Musical Instrument Digital Interface*

OMR – *Optical Musical Recognition*

PPQ – *Pulses per Quarter Note*

RF – Requisito Funcional

RNF – Requisito Não Funcional

SDK - Software Development Kit

UC – *Use Case*

UML – *Unified Modeling Language*

XML – *eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

De acordo com Sá (2001), nos últimos anos a inclusão social tem ganhado espaço seja pela capacitação de profissionais para desenvolver e implantar projetos de educação inclusiva ou através de pesquisas científicas ligadas à educação inclusiva. Porém, apesar da evolução obtida nos últimos anos, pouco se fez pela inclusão dos deficientes visuais nas universidades.

Bonilha (2007) afirma que atualmente, o termo inclusão está associado à integração de pessoas com deficiências físicas ou mentais ao convívio social. Porém, existe uma diferença entre inclusão e integração. De acordo com Almeida (2004), ambas referem-se à inserção de pessoas com necessidades especiais, mas possuem posicionamentos divergentes para a consecução de suas metas. A integração vem dos anos 60 e 70, onde neste modelo os portadores de necessidades educacionais especiais precisavam de tratamento médico e de reabilitação para tornarem-se aptos a realizarem tarefas. A inclusão vem dos anos 80, consolidando-se somente nos anos 90 e consiste em mudar a visão ou padrões da sociedade para tornar-se capaz de acolher as pessoas com necessidades especiais.

De acordo com Tomé (2003), durante muitos anos no século XX o ensino de cegos foi feito em regime de internato, e grande parcela desse ensino resumia-se à música, pois se acreditava ser uma saída profissional para os deficientes visuais.

A notação musical em braile, também denominada musicografia braile, consiste no sistema de leitura e escrita musical convencionalmente adotado por pessoas com deficiência visual. Por meio de partituras musicais em braile, a pessoa cega consegue ler e escrever todos os elementos da grafia musical em tinta. Entretanto, a quantidade de métodos de ensino de música para deficientes visuais é desproporcional à quantidade destes materiais para pessoas que enxergam. Louro (2006) acrescenta que a produção de livros de teoria e de partituras em braile ainda é muito escassa, visto que é alto o custo de edições neste sistema. Esta escassez dificulta a inclusão do aluno deficiente visual em cursos de música.

Diante do exposto, este trabalho propõe o desenvolvimento de uma ferramenta que permita a geração de partituras no sistema de musicografia braile a partir de arquivos no formato MIDI, gerados por um software editor de partituras.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é o desenvolvimento de uma ferramenta que permita a geração de partituras no sistema de musicografia braile a partir de arquivos MIDI.

Este trabalho tem como objetivo específico extrair e interpretar as informações contidas em arquivos no formato MIDI e transcrevê-las para o sistema de musicografia braile.

1.2 ESTRUTURA DO TRABALHO

No capítulo 2 é apresentada a revisão bibliográfica que foi utilizada para dar suporte ao desenvolvimento da ferramenta. Neste capítulo são apresentados conceitos e técnicas referentes à caracterização de congestionamentos e algoritmo de busca de custo mínimo.

O capítulo 3 mostra as etapas de desenvolvimento da ferramenta. Primeiramente são apresentados os requisitos, após isso são mostrados os diagramas utilizados para especificação da ferramenta. Na sequência são descritas as ferramentas e técnicas utilizadas na implementação da ferramenta e sua operacionalidade. Por fim são apresentados os resultados e discussão.

No capítulo 4 é feita a conclusão do trabalho e são apresentadas sugestões para trabalhos futuros.

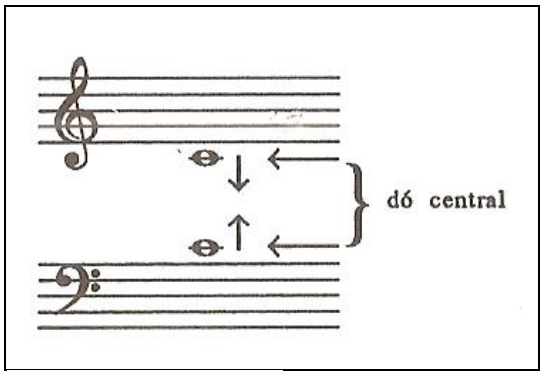
2 FUNDAMENTAÇÃO TEÓRICA

A seção 2.1 trata da teoria musical, bem como sua notação no sistema convencional de escrita e leitura. A seção 2.2 descreve o sistema de escrita e leitura braile e sua utilização para a escrita de partituras para deficientes visuais. A seção 2.3 discorre sobre o formato de arquivo de áudio MIDI. A seção 2.4 descreve os trabalhos correlatos.

2.1 TEORIA MUSICAL

Teoria musical, de acordo com Med (1996), é o conjunto de todos os conhecimentos teóricos em música. Pode ser definida ainda como a estrutura lógica subjacente às práticas musicais. Normalmente, divide-se a teoria musical em áreas de estudo, que variam de acordo com a escola de pensamento. Porém, a maioria destas escolas possui em suas áreas de estudo ao menos análise musical, estética musical e notação musical.

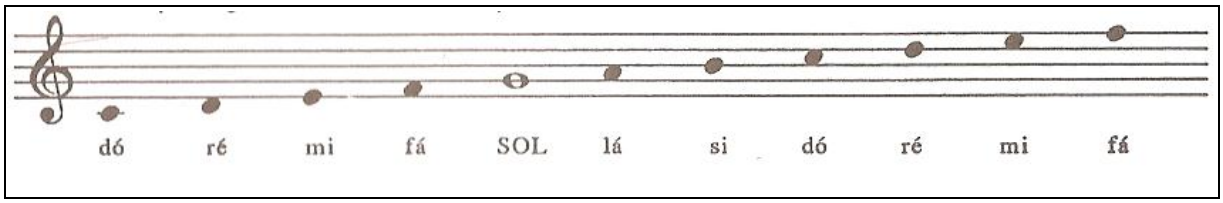
A notação, ou representação gráfica musical é feita em um conjunto de cinco linhas horizontais e paralelas chamado de pentagrama. As notas são escritas sobre as linhas ou entre os espaços, sendo que a altura da nota depende da posição da mesma no pentagrama. As notas mais agudas são escritas nas linhas superiores e as notas mais graves são escritas nas linhas inferiores. Se for necessário, podem ser adicionadas linhas suplementares ao pentagrama, acima ou abaixo, para possibilitar a escrita de outras notas. A altura das notas é definida por um símbolo posicionado na extremidade esquerda chamado de clave. A clave determina ainda a localização da nota que lhe dá nome, sendo assim possível identificar as demais. As claves mais comumente utilizadas são a de sol, utilizada para vozes agudas, e a clave de fá, utilizada para vozes graves, conforme mostra a Figura 1.



Fonte: Med (1996, p. 19).

Figura 1 – As claves de sol (acima), utilizada para sons agudos, e fá (abaixo), utilizada para os sons mais graves

As notas musicais são fundamentalmente sete, sendo elas: dó, ré, mi, fá, sol, lá e si, conforme mostra a Figura 2.

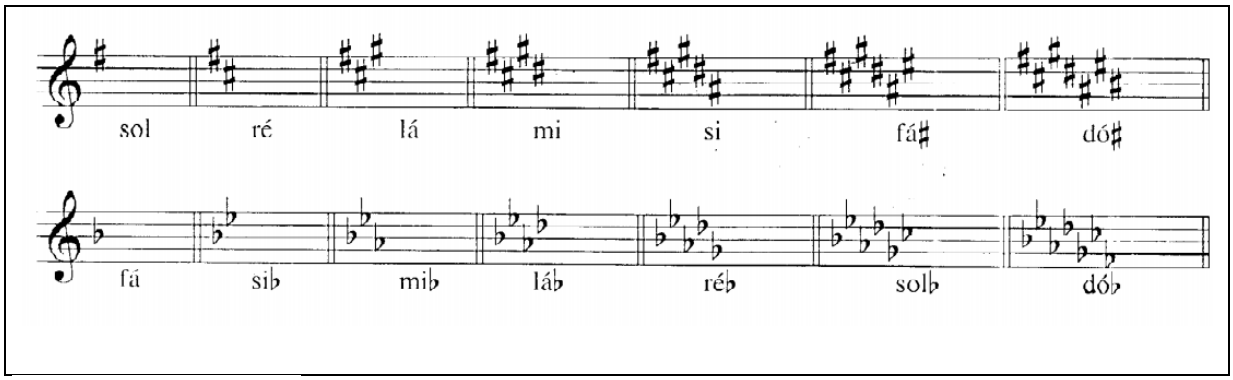


Fonte: Med (1996, p. 16).

Figura 2 - As notas musicais

Algumas destas notas possuem alterações que as aumentam ou diminuem em um semitom, isto é, em meio tom. Estas alterações são chamadas de acidentes e aparecem antes da nota ser escrita na partitura. A alteração de aumento é chamada de sustenido e é representada pelo sinal #. A alteração de diminuição é chamada de bemol e é representada pelo sinal ♭. Estas alterações valem somente para o compasso em que a nota é tocada. Ainda, a alteração vale somente para a nota escrita naquela linha, não afetando as demais oitavas. Tanto o sustenido quanto o bemol podem ter seu valor dobrado, aumentando ou diminuindo, respectivamente em um tom a nota tocada, utilizando os símbolos ✗ e ♭♭. A alteração pode ser cancelada utilizando-se o bequadro, notado com o símbolo ♮. A distância entre duas notas é chamada de intervalo.

O tom de uma música é determinado pela quantidade de sustenidos ou bemóis que são escritos na armadura da clave. A escala do tom deve conter uma nota em cada espaço, não podendo, portanto, haver na mesma armadura sustenidos e bemóis ao mesmo tempo. A Figura 3 mostra os tons e a respectiva quantidade de acidentes.

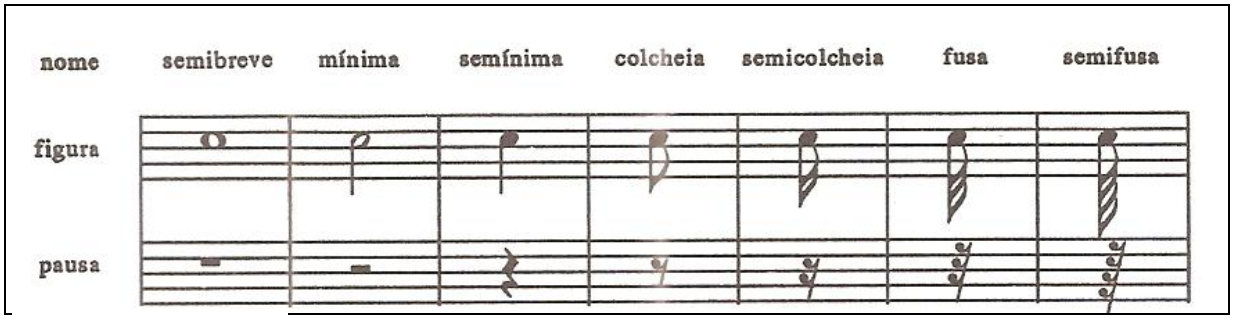


Fonte: Guest (2006, p. 25).

Figura 3 – Quantidade de acidentes por tom

Se o sinal de bequadro aparecer em uma nota alterada por um sinal na armadura da clave, essa será alterada até o final do compasso e somente na linha em que for escrita.

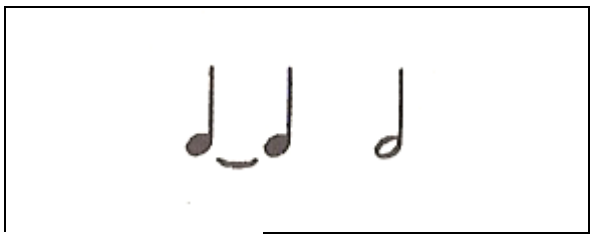
O ritmo de uma música é definido pelas figuras rítmicas, pelo compasso musical e pelo andamento. As figuras rítmicas definem a duração da nota e têm cada uma delas sua pausa correspondente. Cada figura de menor valor equivale à metade do valor de sua antecessora. Por exemplo, duas semínimas equivalem a uma mínima. As figuras são apresentadas na Figura 4.



Fonte: Med (1996, p. 21).

Figura 4 – Figuras rítmicas e suas respectivas pausas

Há casos em que a duração que uma nota deve ser tocada é superior a uma determinada figura rítmica, porém inferior à consecutiva de maior duração. Nesses casos, é possível utilizar ligaduras de duração ou ainda pontos de aumento. A ligadura soma os valores das notas em uma mesma altura que seu sinal abrange. É representada por uma linha curva sobre ou sob a pauta, conforme mostra a Figura 5.

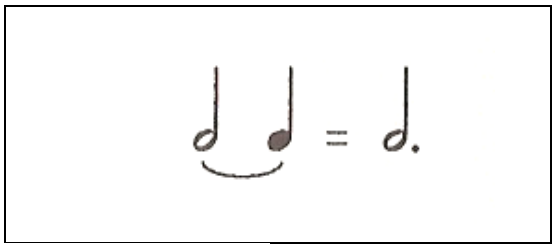


Fonte: Med (1996 p. 47).

Figura 5 – Ligadura de duração

Os pontos de aumento aumentam a duração da nota tocada à metade de sua duração.

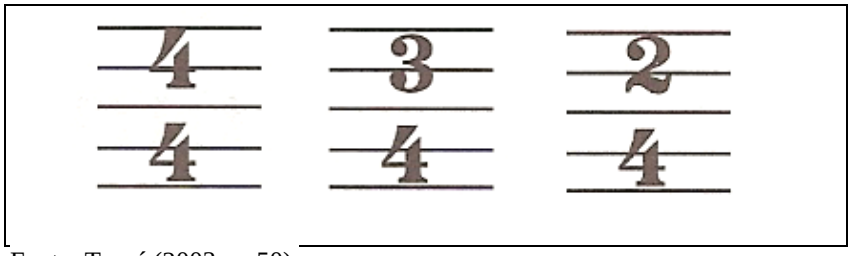
Pode haver ainda notas com dois pontos de aumento, elevando a duração para à metade de seu valor mais ¼ deste valor. O mesmo vale para as pausas, conforme mostrado na Figura 6.



Fonte: Med (1996, p. 39).

Figura 6 – Pontos de aumento

O compasso é uma divisão quantitativa dos sons de uma música, baseada em pulsos e repousos. Os pulsos são as notas tocadas e os repousos são os momentos de silêncio entre uma nota e outra, ou pausas, como denominado na notação musical. A fórmula de compasso é representada logo após a clave por uma fração, conforme mostrado pela Figura 7.



Fonte: Tomé (2003, p. 50).

Figura 7 – Fórmulas de compasso mais comuns

O numerador indica quantas repetições da figura rítmica entram no compasso e o denominador indica a figura que compõe a fórmula do compasso. Por exemplo, a fórmula de compasso 4/4 indica que cada compasso da música é composto pelo valor de quatro semínimas, ou por figuras que totalizem esse valor. Da mesma forma, uma fórmula de compasso ¾ indica que cada compasso é composto por três semínimas.

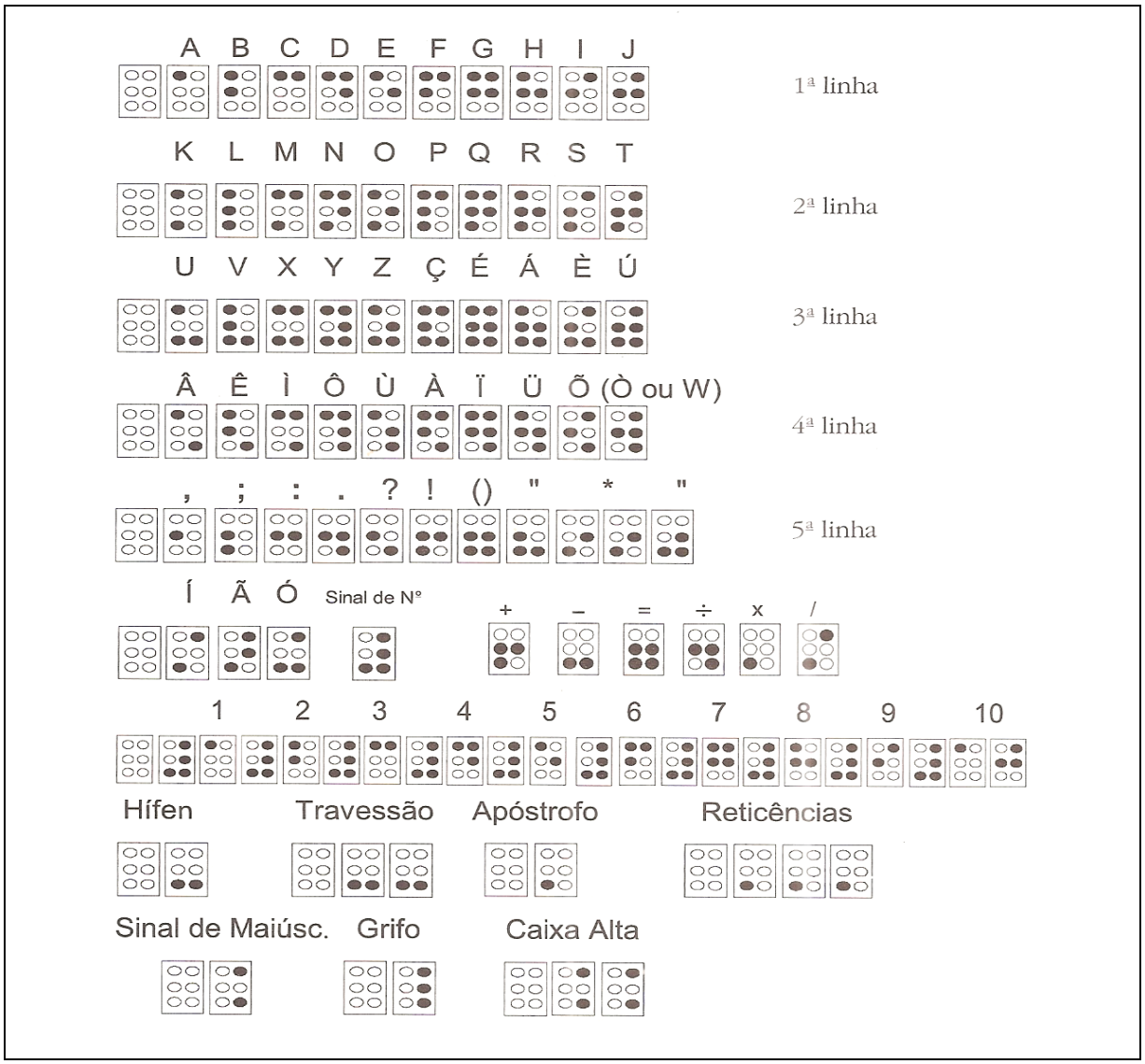
Evidentemente, o ensino de música para deficientes visuais não é possível com o sistema musicográfico aqui apresentado. Contudo, o sistema de escrita braile contempla a escrita musical. Por meio de uma padronização definida e revisada em comitês ao longo dos anos desde a sua criação, é possível escrever partituras como as de tinta, desde músicas simples, até sinfonias e outras formas musicais complexas.

2.2 SISTEMA DE ESCRITA E MUSICOGRAFIA BRAILE

Esta seção foi dividida em duas subseções. A subseção 2.2.1 trata do sistema de escrita braile, explicando a formação dos caracteres. A subseção 2.2.2 trata do sistema de musicografia braile e de como este é utilizado para a escrita de partituras para deficientes visuais.

2.2.1 Sistema de escrita braile

O sistema de escrita braile foi desenvolvido em 1825, pelo educador francês Louis Braile, permitindo que os deficientes visuais tivessem acesso ao universo cultural. Braile, cego desde os três anos de idade, criou um sistema de duas fileiras verticais de três pontos em relevo cada, possibilitando, portanto, 64 símbolos diferentes, dentre letras, numerais e sinais matemáticos, possibilitando sua utilização em campos distintos, como por exemplo, Física, Química e Matemática. A Figura 8 mostra o alfabeto braile.



Fonte: Tomé (2003, p. 37).

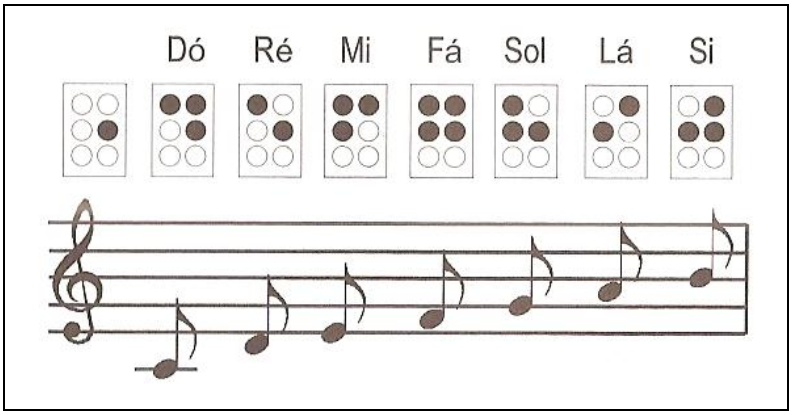
Figura 8 – Alfabeto braile

Os pontos da célula braile são numerados verticalmente, da esquerda para a direita. Os símbolos são organizados em quatro séries de dez. A primeira série de símbolos é formada utilizando somente os pontos 1, 2, 4 e 5. A segunda série de símbolos é formada adicionando o ponto três à célula. A terceira série é formada adicionando os pontos cinco e seis à célula e a quarta série é formada adicionando somente o ponto seis à célula. Os numerais são formados da mesma forma que os símbolos da primeira série, adicionando um símbolo de numeral, formado pelos pontos 3, 4, 5 e 6. Outros sinais gráficos têm sua representação em braile formada pelos 26 símbolos possíveis restantes e variam de acordo com a necessidade de cada língua. Os mesmos símbolos aplicados na escrita braile são utilizados na escrita de música para deficientes visuais.

2.2.2 Sistema de musicografia braile

De acordo com Tomé (2003, p. 23), Louis Braile realizou a sua primeira musicografia em 1829, baseado em seu sistema na obra “Método para escrever as palavras, a música e o cantochão por meio dos pontos”. Propunha um sistema de caracteres musicais baseados nos seis pontos de seu alfabeto. O alfabeto permanece até hoje invariável. Já a musicografia foi modificada pelo próprio Braile ao longo de sua vida e ao longo do tempo foram realizados aperfeiçoamentos no código, de modo que este contemplasse todas as formas de escrita musical. A primeira edição do Novo Manual Internacional de Musicografia Braile foi publicada em inglês em 1996 e teve sua tradução em português lançada em 2004.

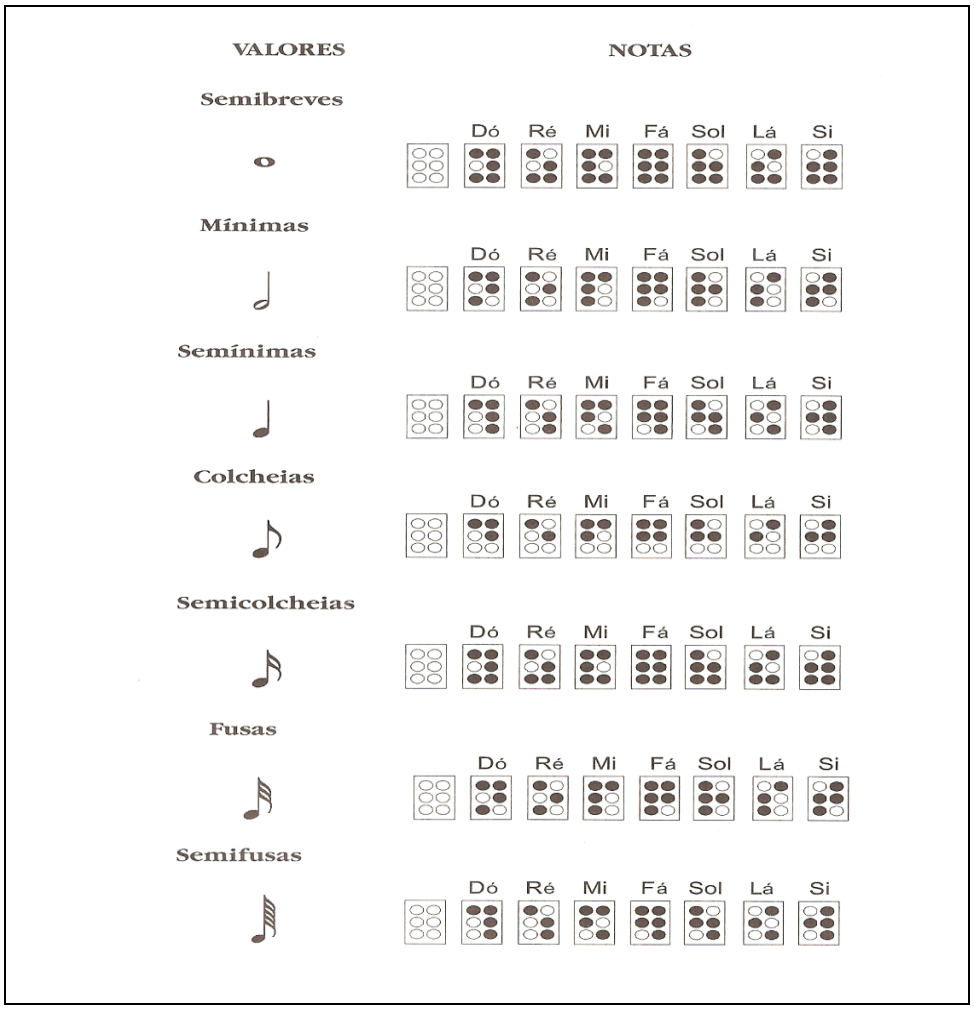
A figura de base da musicografia braile é a colcheia, pois a partir dela são formadas as demais figuras. As figuras de colcheia são formadas utilizando os pontos 1, 2, 4 e 5, conforme mostrado pela Figura 9.



Fonte: Tomé (2003, p. 38).

Figura 9 – Representação das colcheias em braile

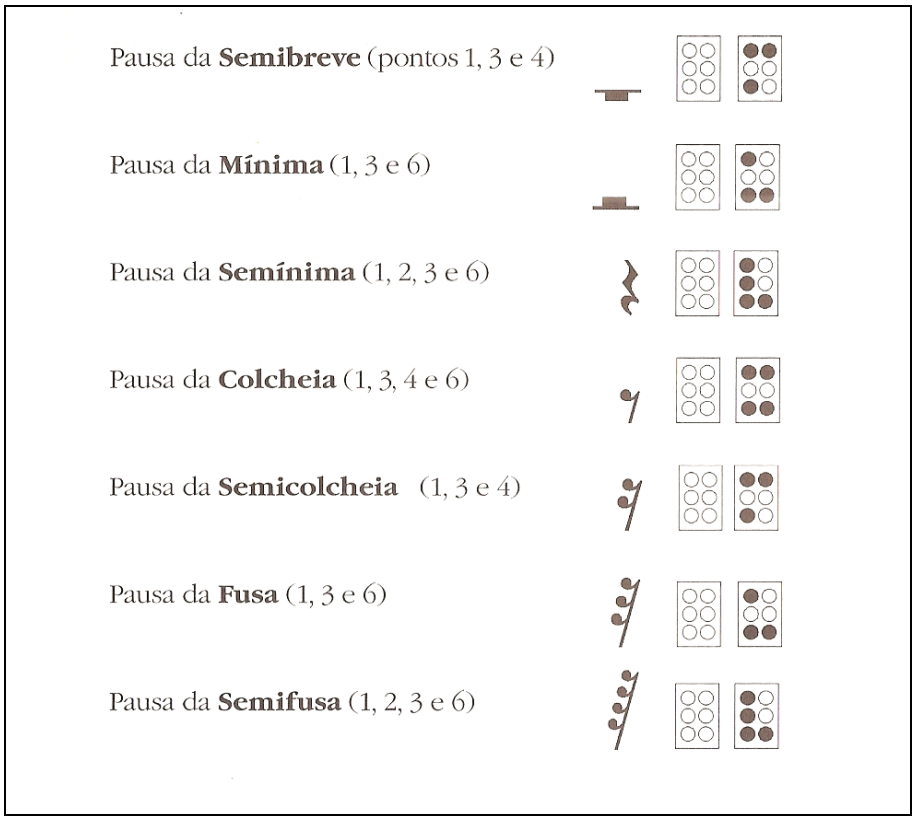
As semínimas e as semifusas são formadas adicionando o ponto seis às figuras de colcheia. As mínimas e as fusas são formadas acrescentando o ponto três. As semibreves e semicolcheias são formadas adicionando os pontos três e seis às figuras de colcheia. Apesar de os mesmos símbolos representarem mais de uma figura rítmica, não há confusão na interpretação, pois o que definirá a figura em questão é a fórmula de compasso. O quadro completo de notas e símbolos é mostrado pela Figura 10.



Fonte: Tomé (2003, p. 40).

Figura 10 – Quadro geral das notas representadas em braile

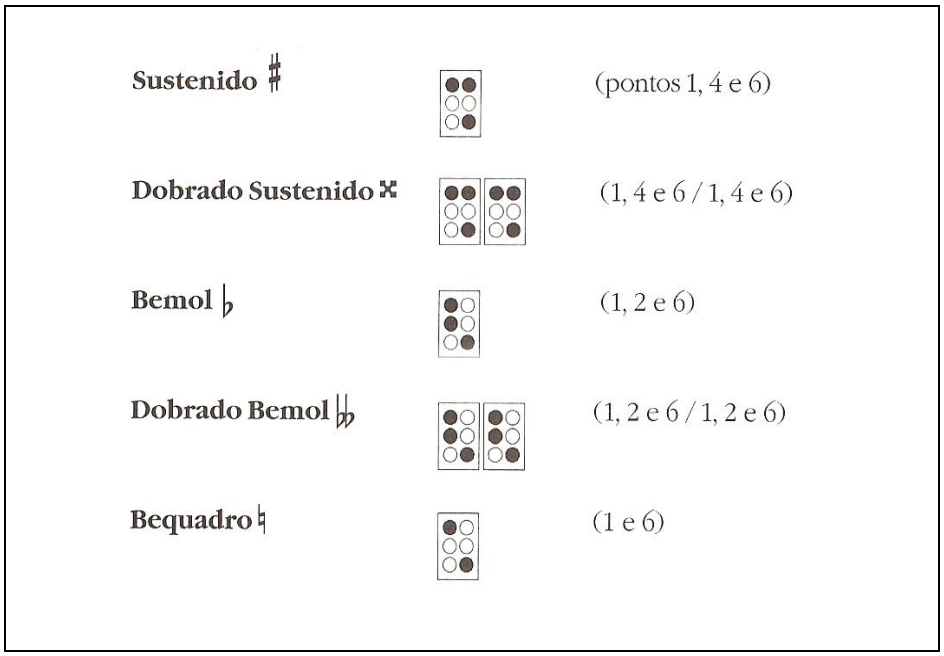
Existem ainda os símbolos de pausa, que seguem a mesma regra de formação das figuras rítmicas, isto é, o mesmo símbolo utilizado para duas figuras distintas. A Figura 11 exibe o quadro de pausas.



Fonte: Tomé (2003, p. 39).

Figura 11 – Quadro de pausas

Os sinais de alteração, também chamados de acidentes, são escritos antes das notas. A Figura 12 mostra estes sinais.

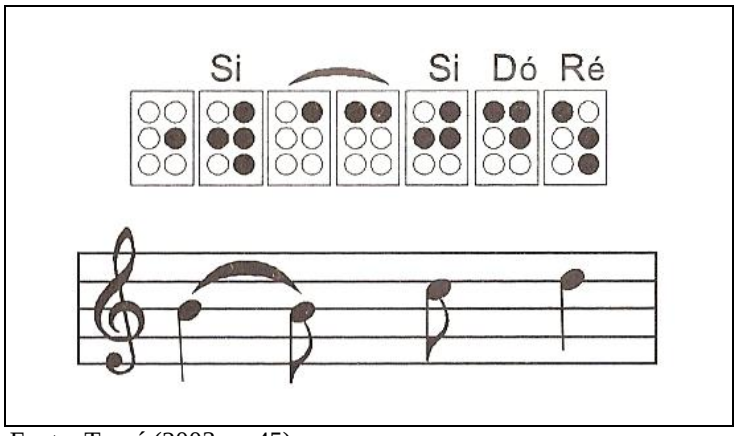


Fonte: Tomé (2003, p. 41).

Figura 12 – Acidentes musicais

As figuras de alteração de duração das notas também tem sua representação no sistema musicográfico braile. A ligadura de duração é representada por dois caracteres: o primeiro

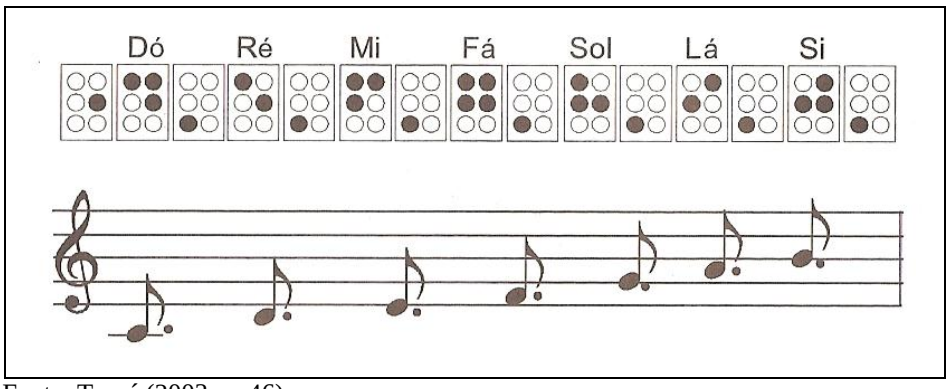
com o ponto quatro e o segundo pelos pontos um e quatro. A Figura 13 mostra a sua utilização.



Fonte: Tomé (2003, p. 45).

Figura 13 – Representação das ligaduras de duração

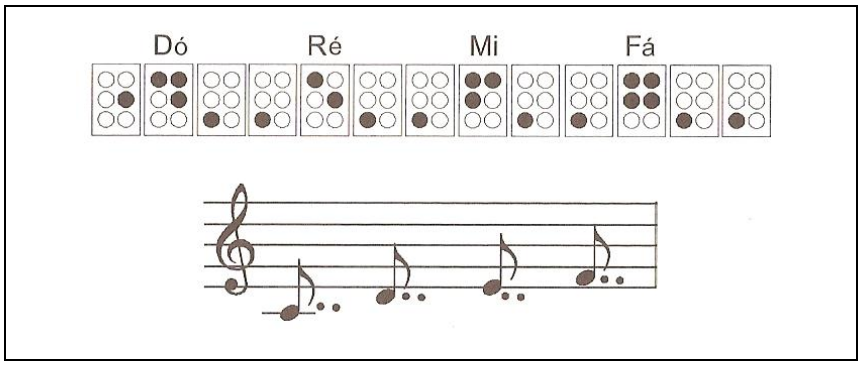
Devemos escrever a nota, seguida pelo símbolo de ligadura e finalmente, novamente a nota. Para a representação dos pontos de aumento, utilizamos um caractere representado pelo ponto três, após a nota, conforme mostra a Figura 14.



Fonte: Tomé (2003, p. 46).

Figura 14 – Representação dos pontos de aumento

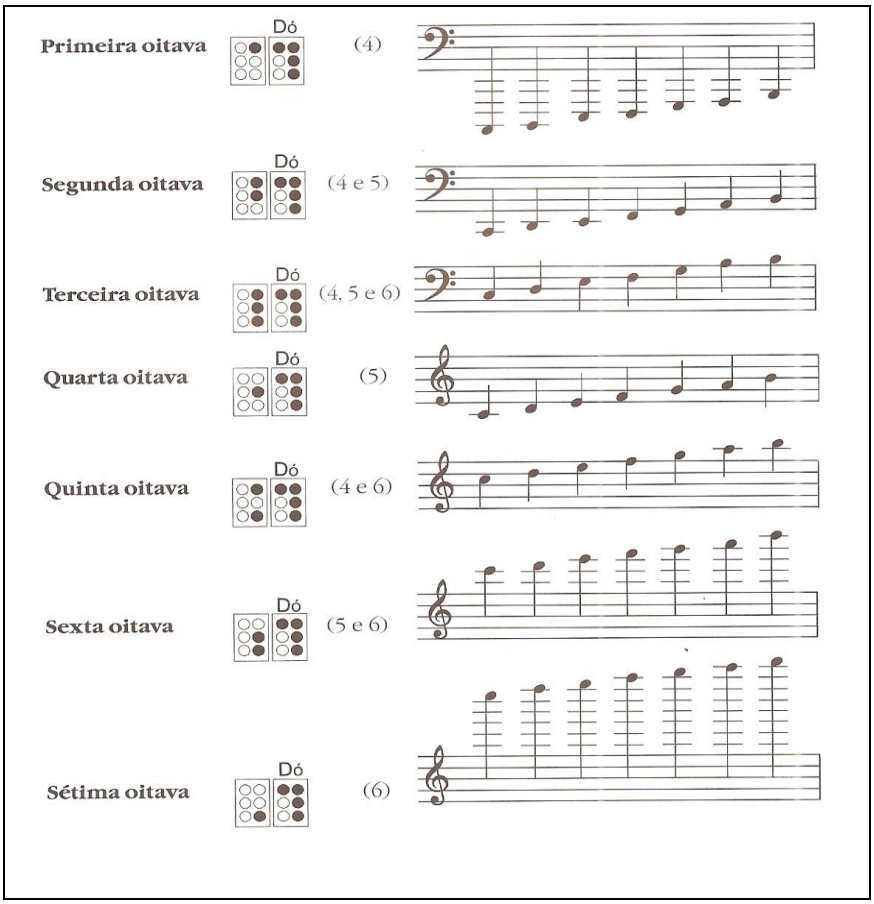
Para a representação do ponto de aumento duplo basta acrescentar um símbolo de aumento, conforme mostra a Figura 15.



Fonte: Tomé (2003, p. 46).

Figura 15 – Representação dos pontos de aumentos duplos

Em uma partitura de tinta é possível identificar a altura de uma nota visualmente, de acordo com sua posição no pentagrama. A notação braile adota um símbolo para identificar oitava em que a nota se encontra. A Figura 16 mostra os símbolos utilizados para as oitavas.



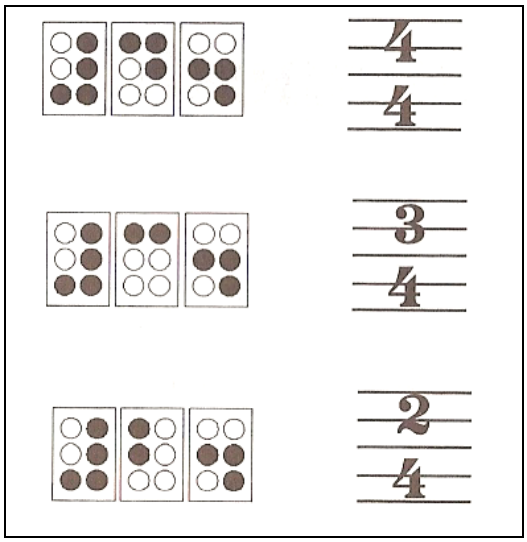
Fonte: Tomé (2003, p. 55).

Figura 16 – Representação das oitavas

Naturalmente, se cada nota escrita recebesse a sua oitava, a partitura ficaria ainda mais extensa do que já é. Por esse motivo existem algumas normas para a utilização dos símbolos de oitava. A primeira nota escrita deve receber o sinal da oitava correspondente. A próxima nota receberá o sinal da oitava caso:

a) as notas estejam distantes por intervalos maiores que de segunda ou terça e a segunda nota ultrapassar a oitava da nota anterior;
b) o intervalo for de sexta ou maior.

Por fim, para representar a fórmula de compasso, elemento importante na divisão rítmica da música, devemos escrever primeiramente o símbolo de numeral (pontos 3, 4, 5 e 6), seguido do numerador e do denominador, escrito nos pontos inferiores da célula braile. A Figura 17 ilustra as fórmulas de compasso mais comuns.



Fonte: Tomé (2003, p. 50).

Figura 17 – Fórmulas de compasso

Os elementos aqui apresentados podem ser representados eletronicamente de diversas formas. Algumas ferramentas de edição de partituras trabalham com um formato chamado MusicXML, porém, todas as principais ferramentas permitem exportar a partitura escrita para o formato MIDI.

2.3 ARQUIVO MIDI

O formato de arquivo MIDI é uma padronização da indústria de instrumentos musicais eletrônicos para permitir a conexão e comunicação entre computadores e aparelhos eletrônicos de forma transparente. O arquivo MIDI é formado por um cabeçalho que identifica informações gerais, como o tipo de formato MIDI, andamento da música e número de faixas do arquivo. O Quadro 1 mostra a estrutura do cabeçalho do arquivo MIDI.

Offset	Tamanho	Tipo	Descrição	Valor
0x00	4	char[4]	ID chunk	"MThd" (0x4D546864)
0x04	4	dword	Tam. chunk	6 (0x00000006)
0x08	2	word	Tipo de formato	0 - 2
0x10	2	word	Nº faixas	1 - 65,535
0x12	2	word	Divisão de tempo	-

Fonte: Sonicspot (2007).

Quadro 1 – Estrutura do cabeçalho do arquivo MIDI

O arquivo é organizado em pedaços de *bytes* chamados de *chunks.* Cada um desses *chunks* é prefixado por oito *bytes*, que indicam o identificador e o tamanho do bloco. Os primeiros quatro *bytes* são sempre a palavra “MThd”, indicando que este é o cabeçalho do arquivo. Os próximos quatro *dwords* informam o tamanho do *chunk*. Na sequência, dois *words* indicam o tipo de formato do arquivo MIDI. De acordo com Maske (2000), o tipo de formato indica a existência de faixas MIDI. Se o tipo for zero, arquivo possui apenas a faixa padrão. O tipo 1 indica que o arquivo possui uma ou várias faixas simultâneas, isto é, que começam a tocar ao mesmo tempo. O tipo 2 indica que o arquivo possui uma ou diversas faixas sequencialmente independentes, isto é, que não começam a tocar ao mesmo tempo. Em seguida é indicado o número de faixas que vêm na sequência do cabeçalho. Por fim, a divisão de tempo indica como o relógio interno do MIDI será decodificado em segundos.

Caso haja faixas no arquivo, estas vêm logo após o cabeçalho. O cabeçalho do bloco referente às faixas é formado pelo identificador e o tamanho do *chunk*, seguido pelos eventos MIDI. O Quadro 2 exibe a estrutura do cabeçalho do bloco de faixas.

Offset	Tamanho	Tipo	Descrição	Valor
0x00	4	char[4]	chunk ID	"MTrk" (0x4D54726B)
0x04	4	dword	Tamanho chunk	-
0x08	Eventos Midi			

Fonte: Sonicspot (2007).

Quadro 2 – Cabeçalho do bloco de faixas do arquivo MIDI

O cabeçalho de uma faixa MIDI sempre começa com a palavra “MTrk” e na sequência, um *dword* define o tamanho do *chunk*. O resto do *chunk* é completado por eventos MIDI. Os eventos contêm mensagens, que são interpretadas pelo controlador MIDI e dizem o que deve ser feito. Existem mensagens para iniciar e interromper a execução de uma nota, enviar textos, controlar dispositivos, como por exemplo, pedais de expressão para teclados eletrônicos, dentre muitos outros. Essas mensagens são formadas por um vetor de *bytes*, contendo o tempo interno do MIDI, o tipo da mensagem, o seu tamanho e finalmente, a mensagem. Cada tipo de mensagem tem um vetor de tamanho diferente, dependendo dos dados que ele envia.

A especificação do arquivo MIDI utiliza um tempo interno para permitir a sincronização de seus eventos, isto, é, para que as mensagens sejam executadas por todos os dispositivos em que estiver sendo usado simultaneamente. Esse tempo é medido em pulsos por semínima, (PPQ). Segundo Prado (2006, p.18), quanto maior a quantidade de pulsos, maior a precisão do sequenciador MIDI. Normalmente, esses equipamentos trabalham com medições de 480 a 960 PPQs. Para os propósitos desta ferramenta, somente são utilizados os

eventos de manipulação de notas e textos. As mensagens de texto são irrelevantes para a identificação das notas, porém, servem para armazenar o título e o compositor do MIDI.

Para iniciar a execução de uma nota o arquivo MIDI envia um evento do tipo `NOTE_ON`. Este evento contém, além do identificador do evento, `00x9`, o canal MIDI em que o evento ocorrerá, a nota que será executada, representada por um número de 0 a 128, e a intensidade, ou volume com que a nota será executada.

Para interromper a execução de uma nota o arquivo MIDI envia uma mensagem do tipo `NOTE_OFF`. Esta mensagem possui os mesmos parâmetros da mensagem `NOTE_ON`, diferenciando-se apenas pelo identificador da mensagem, que é `00x8`. Alguns programas editores de partitura não geram arquivos MIDI com eventos de `NOTE_OFF`, mas enviam eventos de `NOTE_ON` atribuindo zero ao parâmetro de intensidade.

Alguns eventos não são comandos para executar alguma ação no controlador MIDI, porém são enviadas no arquivo mesmo assim. Esses eventos são chamados de meta eventos e são utilizados para enviar informações de texto e relativas à fórmula de compasso, tom da música, etc. Sua estrutura é mostrada no Quadro 3.

Nome	Tamanho	Valor
Status byte	1 byte	Sempre 0xFF
Tipo do evento	1 byte	Variável
Tamanho	1 byte	0-255
Dados	Variável	-

Fonte: Sonicspot (2007).

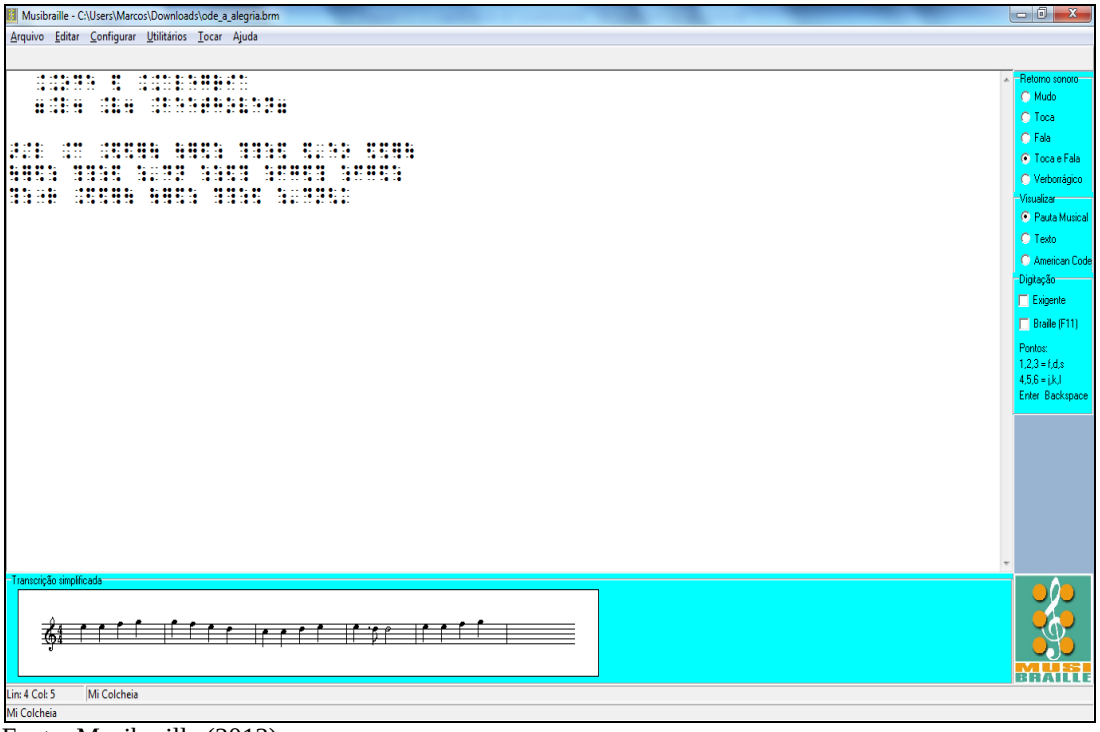
Quadro 3 – Estrutura do meta evento

A estrutura do meta evento é o identificador de meta mensagem, 0xFF, seguido pelo identificador do tipo de mensagem, que pode variar, conforme a mensagem enviada. Para textos em geral, como o título de uma música, é usado o *byte* 0x01. Para os nomes das faixas que compõem o arquivo MIDI é usado o *byte* 0x03. A fórmula de compasso é indicada pelo *byte* 0x58. O tom de uma música é definido pelo *byte* `0x59`. Em seguida, o tamanho da mensagem é definido e por último, é enviada a mensagem propriamente dita.

2.4 TRABALHOS CORRELATOS

Existem poucos softwares disponíveis para a geração de partituras no sistema de musicografia braile. Uma destas soluções é o projeto Musibraille (INSTITUTO BENJAMIN CONSTANT, 2012). Há ainda ferramentas proprietárias, como o software Goodfeel (DANCING DOTS, 2005) e o Tocatta (PENTRONICS AND OPTEC SYSTEMS, 2002).

De acordo com o *site* oficial, o projeto Musibraille destina-se a criar condições favoráveis à aprendizagem musical das pessoas com deficiência visual que sejam equivalentes às dos colegas de visão normal. O projeto é mantido pelo Instituto Benjamin Constant, uma instituição tradicional para ensino de deficientes visuais no Brasil. O software possui dois modos de entrada de notas musicais pelo teclado. É possível digitá-las normalmente, utilizando as letras correspondentes a cada nota no código musicográfico. Também é possível digitar as notas como em uma máquina de escrever braile Perkins[1], permitindo assim, que deficientes visuais possam utilizar o software e editar suas próprias partituras. Neste caso, existe uma assistência por áudio para ajudar o usuário. A Figura 18 mostra a tela principal do Musibraille.



Fonte: Musibraille (2012).

Figura 18 – Tela principal do Musibraille

[1] Máquina de escrever braile. É composta por seis teclas correspondentes aos pontos da célula braile, mais a tecla de espaço.

O Musibraille permite executar as partituras escritas, facilitando assim a verificação do trabalho feito pelo próprio usuário. Também é possível a geração e impressão da versão em tinta da partitura, o que possibilita que pessoas não deficientes também possam ter acesso à peça musical escrita. O programa conta ainda com um dicionário de acordes em braile e uma ferramenta de ditado musical, que permite o aprendizado do sistema de musicografia braile. Além disso, o software possui assistência por voz, facilitando assim seu uso por deficientes visuais.

O software Goodfeel, desenvolvido pela empresa Dancing Dots (2005) é uma solução paga composta por três softwares: SharpEye, Lime e Goodfeel. O software SharpEye é utilizado para digitalizar partituras em tinta, convertendo-as para o formato MIDI. No processo de conversão da partitura para o formato MIDI podem ocorrer imperfeições, que são corrigidas utilizando o Lime. Por fim, a conversão do arquivo MIDI em braile musical é feita pelo software Goodfeel. Com este software é possível transcrever solos vocais, partes instrumentais e até mesmo orquestrações completas. A Figura 19 mostra a tela principal do Goodfeel.

Fonte: Dancing Dots (2005).

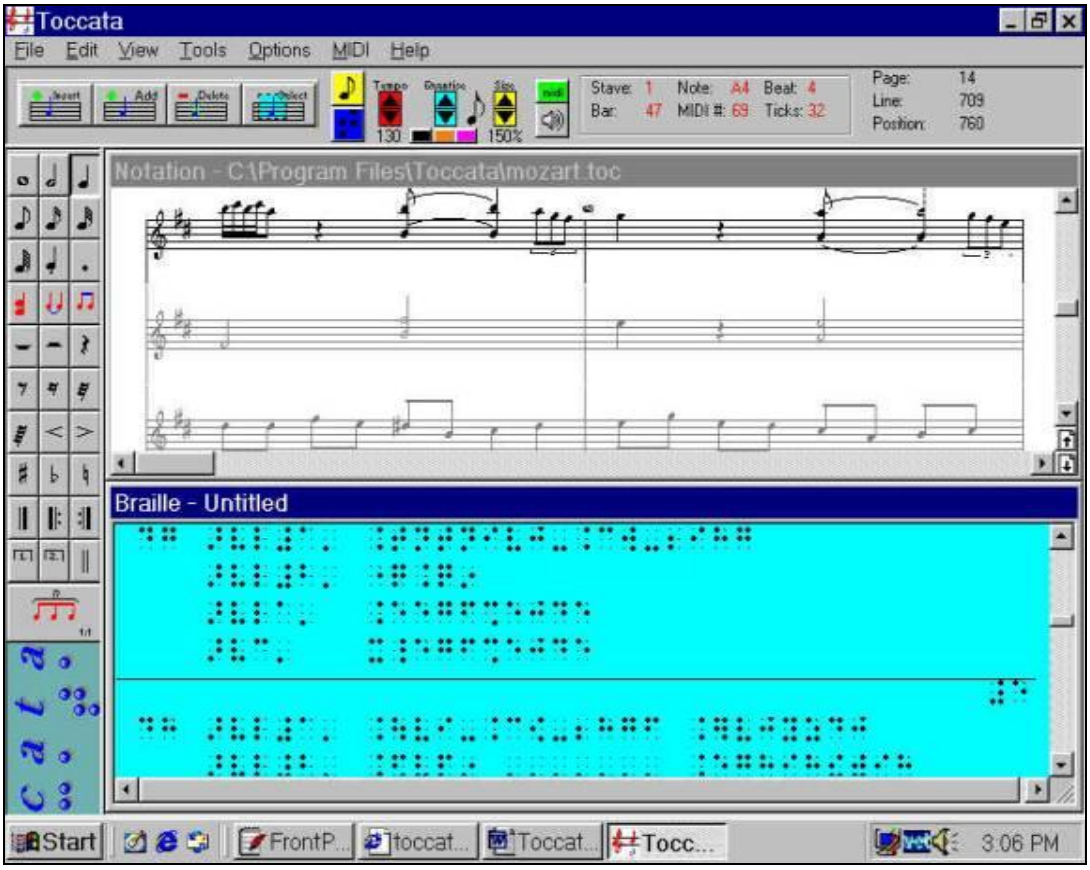
Figura 19 – Tela principal do Goodfeel

Este software permite ainda que sejam importados arquivos gerados no editor de partituras Sibelius no formato MusicXML[2], por meio de um *plug-in*.

Toccata, desenvolvido pela empresa Pentronics and Optek Systems (2002), é um software de

[2] De acordo com o *site* oficial, MusicXML é um formato de arquivo de notação musical aberto, baseado em XML. Permite o intercâmbio de formatos digitais de partitura.

tradução para braile musical que utiliza arquivos *Notation Interchange Format File* (NIFF), além dos arquivos MIDI. Existe também uma versão para pessoas com baixa visão, Magni-CCata, que é distribuída junto com o Toccata. A Figura 20 mostra uma tela do Toccata.



Fonte: Pentronics and Optec Systems (2002).

Figura 20 – Tela principal do Toccata

No Toccata é possível editar a partitura escrevendo-a no editor, ou utilizando o teclado do computador como uma máquina Perkins. O software permite ainda a conversão de partituras no formato de tinta por meio OMR utilizando o software Sharpeye2.

O Quadro 4 apresenta um comparativo das caraterísticas destes softwares.

Características / trabalhos correlatos	Musibraile (2012)	Toccata (2002)	Goodfeel (2005)
assistência de usabilidade por áudio	X	-	-
dois modos de entrada (Perkins e teclado padrão)	X	X	-
importação de MIDI	-	X	X
importação de MusicXML	X	-	X
permite a edição de partitura padrão	-	X	-
Custo	gratuito	US$ 795	US$ 1395

Quadro 4 – Características dos trabalhos relacionados

No Quadro 4 é possível verificar que o Musibraille e o Toccata permitem a entrada da partitura de duas formas: utilizando o teclado normalmente, digitando cada nota, ou utilizando-o como uma máquina Perkins. Tocata não permite a importação de arquivos do tipo MusicXML, operação permitida no Musibraille e Goodfeel.

O projeto Musibraille permite a edição de partituras pelo próprio deficiente visual, bem como por usuários sem a deficiência. Entretanto, somente é possível editar partituras no formato braile, embora seja possível visualizar simultaneamente a partitura em formato de tinta. A suíte de programas Goodfeel permite a geração de partituras em formato braile a partir de arquivos MIDI. Toccata também permite a importação de arquivos MIDI, bem como gerar a partitura braile a partir de uma digitalização da partitura de tinta. Para ambos os softwares é necessário o software SharpEye, que faz a conversão da partitura para o formato MIDI.

O projeto Musibraille é o único software gratuito disponível na área de musicografia braile e não contempla a importação de arquivos MIDI. As ferramentas disponíveis que realizam este trabalho são soluções caras e por consequência, pouco acessíveis. Esse é um fator que dificulta a adoção destas ferramentas em larga escala no ensino superior de música no Brasil. A ferramenta proposta neste trabalho visa suprir esta deficiência permitindo assim a importação de arquivos MIDI para a geração de partituras braile.

3 DESENVOLVIMENTO

Neste capítulo são abordadas atividades relacionadas ao projeto e desenvolvimento da ferramenta proposta neste trabalho.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

A ferramenta deverá:

a) permitir a leitura de arquivos MIDI (Requisito Funcional – RF);
b) permitir a seleção de pistas[3] de áudio do arquivo MIDI para geração da partitura (RF);
c) permitir a geração de partituras no sistema de musicografia braile a partir do arquivo MIDI (RF);
d) permitir o armazenamento e a impressão da partitura braile gerada (RF);
e) ser implementado utilizando a linguagem de programação Java (Requisito Não-Funcional – RNF);
f) ser implementado utilizando a biblioteca Java Sound (RNF);
g) ser implementado utilizando o ambiente de desenvolvimento Eclipse (RNF).

3.2 ESPECIFICAÇÃO

A especificação da ferramenta foi desenvolvida seguindo a análise orientada à objetos, utilizando a *Unified Modeling Language* (UML) em conjunto com a ferramenta *Enterprise Architect* para o desenvolvimento dos diagramas de caso de uso, de classe e de sequência.

[3] As pistas de um arquivo MIDI podem ser entendidas como os canais de uma mesa de som, cada um contendo o áudio de um instrumento que compõe a música. Desta forma o arquivo MIDI pode ser interpretado por diversos aparelhos, permitindo a sua interoperabilidade.

3.2.1 Diagrama de casos de uso

A Figura 21 exibe o diagrama de casos de uso para a ferramenta.

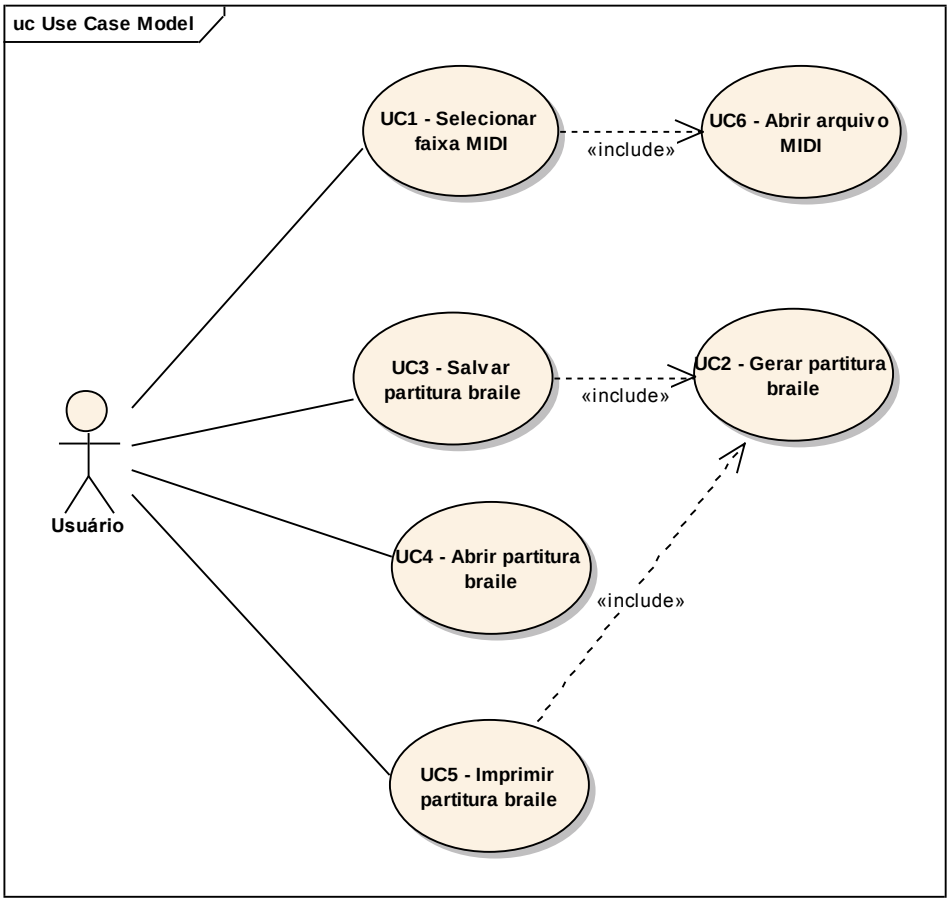


Figura 21 – Diagrama de casos de uso

No caso de uso `UC 1 – Selecionar faixa MIDI`, o usuário selecionará uma faixa do arquivo MIDI para que a partitura seja gerada. Para isso, o arquivo MIDI deve ser aberto conforme especificado no caso de uso `UC6 – Abrir arquivo MIDI`.

No caso de uso `UC 2 – Gerar partitura braile`, o usuário gera a partitura braile a partir do arquivo MIDI previamente aberto.

No caso de uso `UC 3 – Salvar partitura braile`, o usuário salva a partitura braile para uso posterior. Para isso, é necessário gerar a partitura braile, conforme descrito no caso de uso `UC 2 – Gerar partitura braile`.

No caso de uso `UC 4 – Abrir partitura braile`, o usuário abre uma partitura braile previamente salva.

No caso de uso `UC 5 – Imprimir partitura braile`, o usuário imprime a partitura braile gerada. Para isso, é necessário gerar a partitura braile, conforme descrito no

caso de uso `UC 2 – Gerar partitura braile`.

3.2.2 Diagrama de classes

A estrutura de classes da ferramenta está dividida em dois pacotes, um de classes básicas que estruturam a ferramenta, e outro de classes que realizam o processamento dos arquivos MIDI e sua tradução para a notação musicográfica braile.

3.2.2.1 Diagrama de classes básicas

A Figura 22 apresenta o diagrama das classes básicas que compõem a ferramenta. Trata-se dos elementos comuns às partituras e métodos auxiliares no processo de identificação das notas.

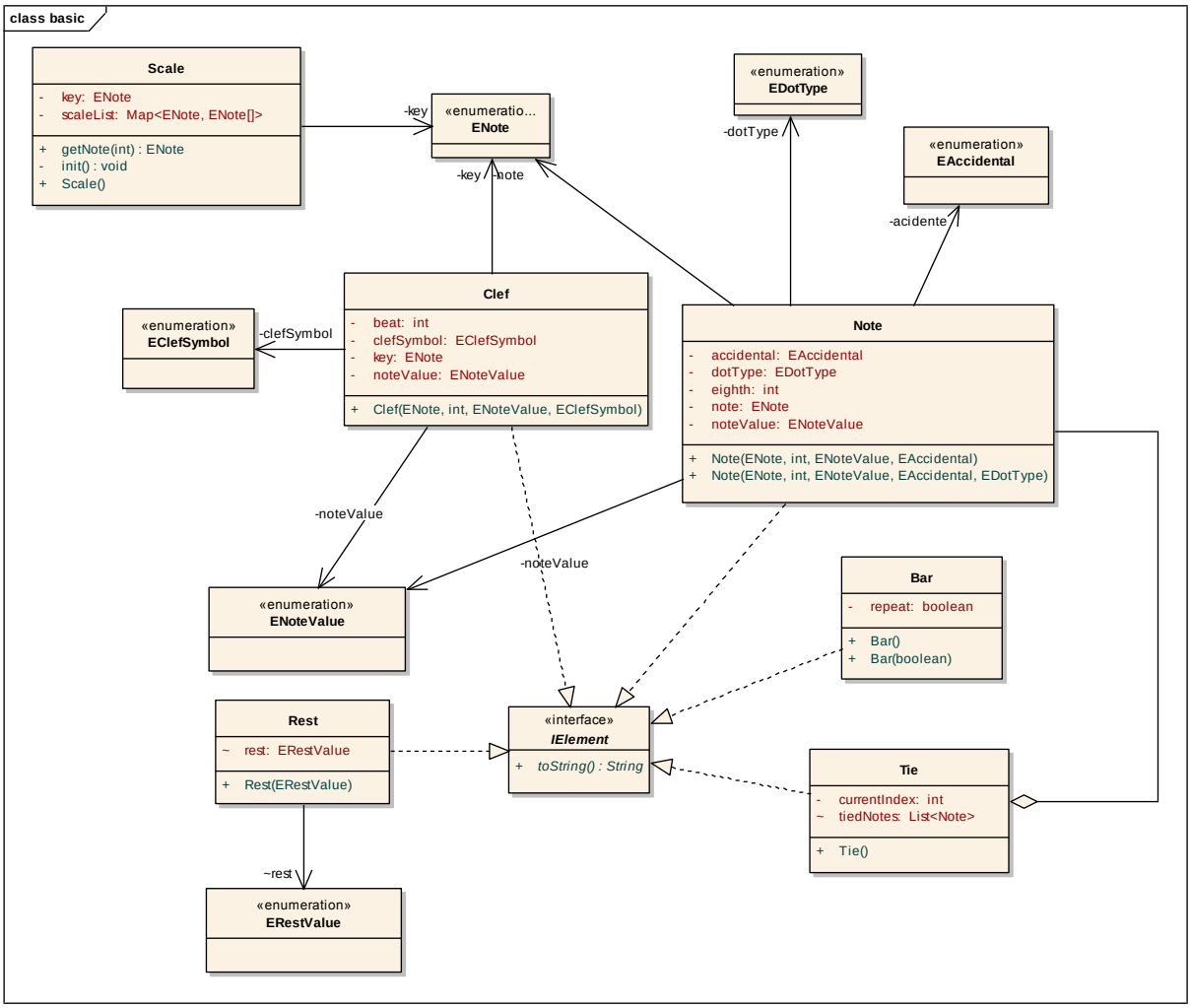


Figura 22 – Diagrama de classes básicas

Uma partitura em tinta possui diversos elementos, dentre eles, as notas, as pausas, a clave e a barra de divisão de compasso. Estes elementos devem ser representados também na partitura braile. Para isso existe a interface `IElement`, que não possui quaisquer métodos, pois apenas serve de elemento comum entre os componentes da partitura.

A classe `Clef` representa a armadura de clave da partitura, contendo informações referentes à clave e a formula de compasso. Também armazena informações referentes ao andamento musical.

A classe `Note` representa as informações de cada nota da música. Estas informações são o nome da nota, a oitava em que a nota é executada, a duração, representada pela enumeração `ENoteValue`, o tipo de acidente, representado pela enumeração `EAccidental`.

A classe `Rest` representa as pausas da música. Somente possui a informação da duração da pausa, representada pela enumeração `ERestValue`.

As divisões de compasso são representadas pela classe `Bar`. Esta classe não possui métodos, pois o processamento da divisão de compasso é feito no momento do processamento

do arquivo MIDI.

Ainda que haja uma mensagem que informe o tom da música, esta não é relevante na execução do arquivo MIDI, sendo apenas informativa. Quando o arquivo MIDI é lido somente suas notas são tocadas, não importando, portanto o tom. Porém, isto é necessário para processar corretamente a armadura da clave e as notas que serão escritas na partitura. Este processamento é realizado pela classe `Scale`.

A classe `Tie` trata das ligaduras de duração, mantendo a lista das notas que compõem a ligadura. Este símbolo é usado quando não é possível representar a prolongação da duração de uma nota utilizando pontos de aumento. É possível em uma partitura de tinta que uma nota seja escrita utilizando simultaneamente pontos de aumento e ligadura. Neste caso, será utilizada somente a ligadura representando integralmente o valor.

É importante ressaltar que este modelo representa os elementos de uma música e não a própria música. A música será representada no momento do processamento do arquivo MIDI.

3.2.2.2 Diagrama de classes envolvidas no processamento do arquivo MIDI

O processamento do arquivo MIDI e sua conversão para a notação da musicografia braile está especificado no diagrama da Figura 23.

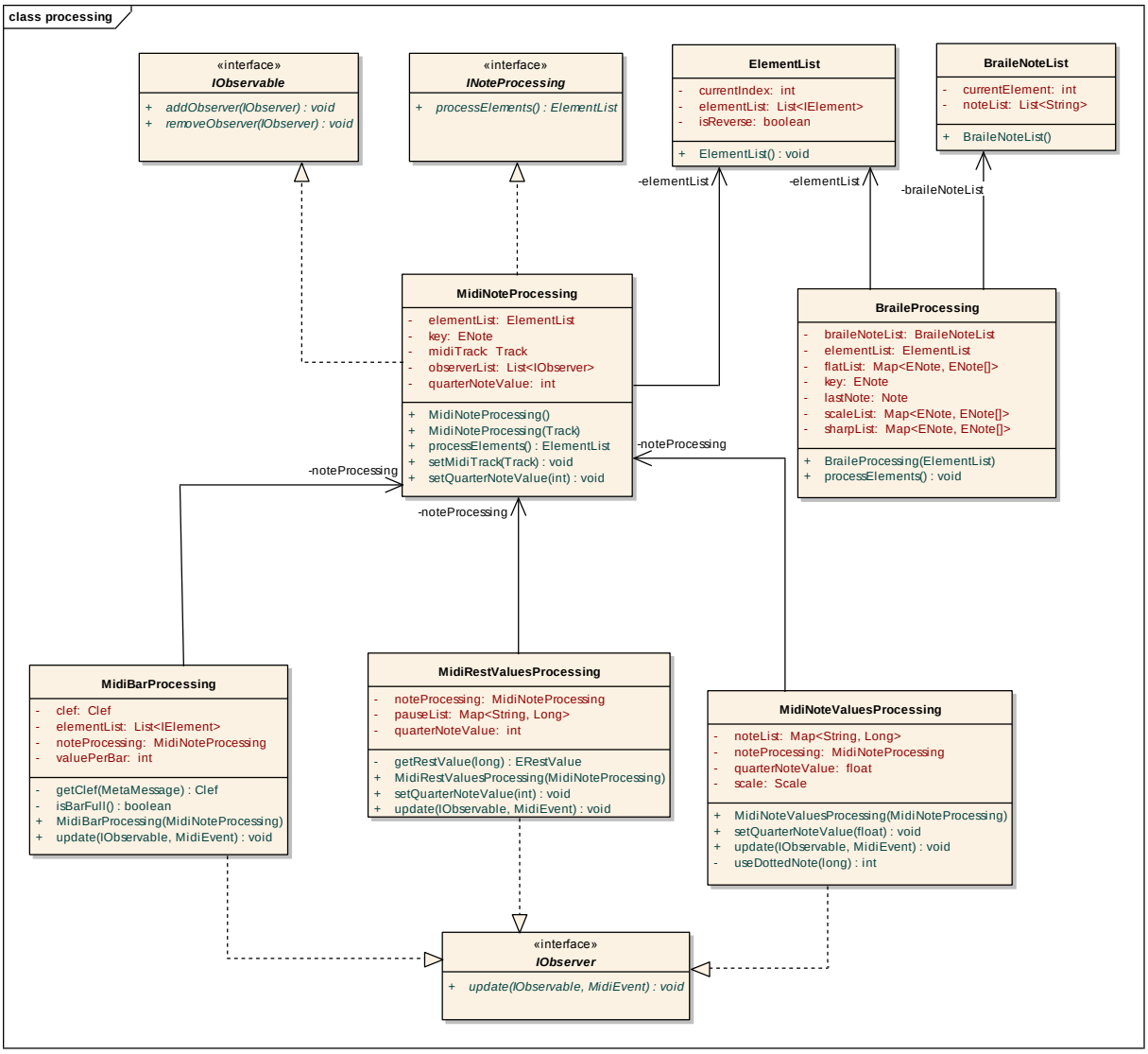


Figura 23 – Diagrama de classes de processamento

Visando a extensão deste trabalho com, por exemplo, a leitura de arquivos MusicXML, a implementação do processamento do arquivo MIDI é realizada por uma classe que estende a interface `INoteProcessing`.

O processamento do arquivo MIDI é realizado pela classe `MidiNoteProcessing`. Esta classe separa o processamento entre notas, implementado pela classe `MidiNoteValuesProcessing`, pausas, implementado pela classe `MidiRestValuesProcessing` e barras de compasso, clave e demais sinais, implementado pela classe `MidiBarProcessing`. Esta estrutura implementa o padrão de projetos *Observer*, onde `MidiNoteProcessing` é o observador e as demais classes são os observados.

A classe `ElementList` é a responsável pelo armazenamento e manipulação dos elementos processados no arquivo MIDI. Também é a entrada necessária para a conversão no sistema musicográfico braile.

O processamento e tradução da lista de elementos é realizada pela classe `BrailleProcessing`. Nesta classe os elementos previamente processados do arquivo MIDI serão lidos um a um e traduzidos para o sistema musicográfico braile. Contém também métodos de correção das notas enviadas pelo arquivo MIDI para correção de escala, para escrever a partitura corretamente. O resultado desta conversão é mantido pela classe `BraileNoteList`.

3.2.3 Diagrama de sequência

Os diagramas de sequência mostram como os grupos de objetos colaboram em algum comportamento ao longo do tempo registrando assim as trocas de mensagens entre os objetos envolvidos em um caso de uso. A seguir é mostrado o diagrama da tradução do arquivo MIDI para braile.

3.2.3.1 Processamento do arquivo MIDI e tradução para braile

A Figura 24 mostra o diagrama de sequência que permite visualizar o caso de uso `UC 2 - Gerar Partitura Braile`, que foi descrito na seção 3.2.1.

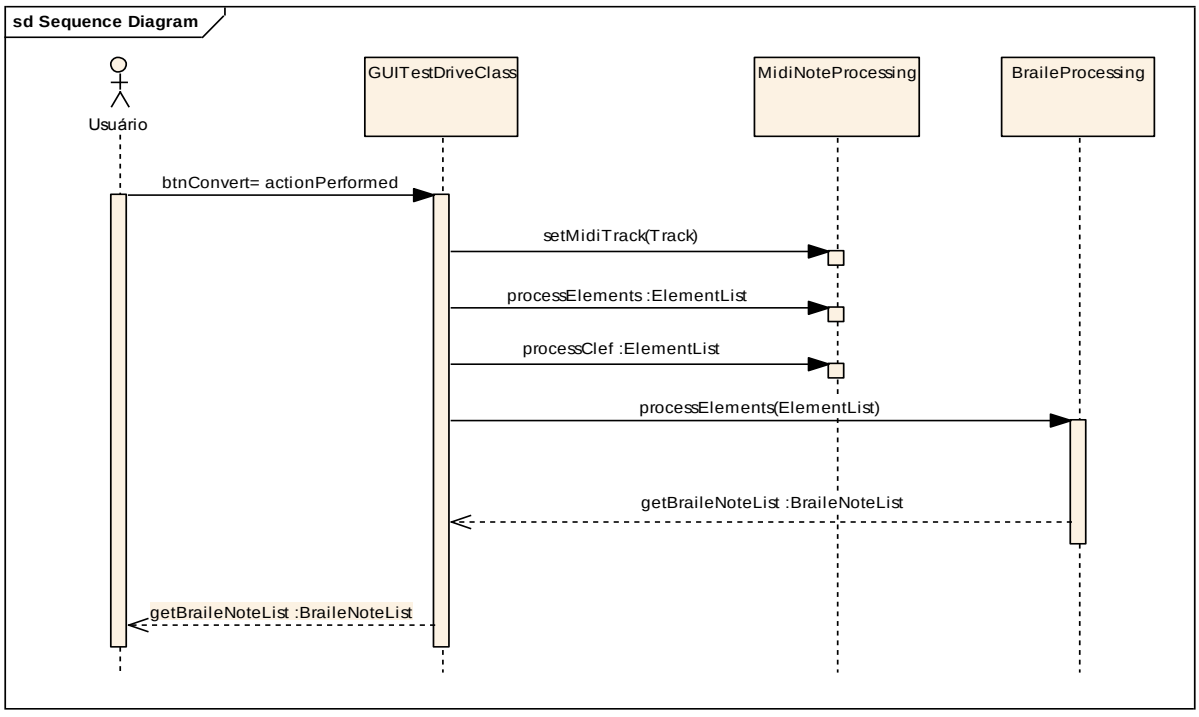


Figura 24 – Diagrama de sequência de tradução e processamento do arquivo MIDI para braile

Para que seja possível converter um arquivo MIDI é necessário que este seja carregado previamente pelo usuário. Quando o usuário pressiona o botão *Converter*, a ferramenta primeiramente processa o arquivo MIDI, determinando a faixa MIDI selecionada para gerar a partitura. Em seguida, são identificadas as notas, pausas e demais elementos que compõem a sequencia MIDI, utilizando para isso o método `processElements()`, da classe `MidiNoteProcessing`. O próximo passo é o processamento da clave, de sol ou de fá, realizado pelo método `processClef()`, da classe `MidiNoteProcessing`. O reconhecimento da clave de dó não foi implementado. Logo após isso, é chamado o método `processElements()`, da classe `BraileNoteProcessing`. Neste momento é realizada a tradução de cada símbolo do arquivo MIDI previamente processado. O retorno do processamento é uma lista de pequenos strings que representam os símbolos utilizados na musicografia braile. O ultimo passo é a exibição do resultado na tela, obtido pela chamada do método `getBraileNoteList()`.

3.3 IMPLEMENTAÇÃO

A seguir são mostradas as técnicas e ferramentas utilizadas e a operacionalidade da implementação.

3.3.1 Técnicas e ferramentas utilizadas

Para o desenvolvimento da ferramenta foi utilizada a linguagem de programação Java no ambiente de desenvolvimento Eclipse. Para o processamento dos arquivos MIDI foi utilizada a biblioteca Java Sound.

3.3.1.1 Processamento do arquivo MIDI.

A conversão do arquivo MIDI para o sistema de musicografia braile começa com o processamento do arquivo, identificando as notas e demais elementos musicais. Para a

implementação do algoritmo foi adotado o padrão de projetos *Observer*, sendo o processo disparado pela classe `MidiNoteProcessing`, a classe observadora, através do método `processElements`. Esta classe implementa ainda a interface `INoteProcessing`, tendo em vista a possibilidade de escrever outras fontes de processamento para tradução para braile, como os arquivos MusicXML, por exemplo.

O algoritmo separa o processamento das notas, realizado pela classe `MidiNoteValuesProcessing`, das pausas, realizado pela classe `MidiRestValuesProcessing` e das divisões de compasso e armadura de clave, implementado pela classe `MidiBarProcessing`. Essas são as classes que o observador deve notificar.

O processamento do arquivo começa com a seleção de uma das faixas do arquivo MIDI. Esta faixa é representada pela classe `Track`, da biblioteca Java Sound. Antes de iniciar a interpretação é necessário informar o valor definido para a semínima e adequar a faixa MIDI. Este valor é retornado pelo método *getResolution,* da classe *Sequence*. Em seguida é necessário adequar a faixa MIDI selecionada, atribuindo zero para o parâmetro `velocity` da mensagem MIDI onde ocorrer uma mensagem `NOTE_OFF`. Esta adequação é feita chamando o método `prepareTrack`. É possível acessar cada evento constante na faixa por meio do método `get`, passando seu o índice. Este método retorna um objeto do tipo `MidiEvent`, que contem a mensagem MIDI. Este evento é enviado para todos os observadores, que processarão e interpretarão a mensagem. Este processo é executado pelo método `loadElements`, exibido pelo Quadro 5.

```
01        private void loadElements() {
02              for (int i = midiTrack.size() - 1; i >= 0; i--) {
03                    for (IObserver o : observerList) {
04                          o.update(this, midiTrack.get(i));
05                    }
06              }
07        }
```

Quadro 5 – Método principal do processamento do arquivo MIDI

É importante notar que as mensagens são lidas na ordem contrária de sua listagem, isto é, da última mensagem para a primeira. Isto ocorre porque a marcação de tempo interno do arquivo MIDI, enviada com cada mensagem, é essencial para que seja possível identificar a duração das notas e pausas e consequentemente, os compassos. As classes observadas são notificadas pelo método `update` (linha 04).

O processamento das notas é feito lendo as informações do evento recebido. Se neste evento estiver contido um objeto `ShortMessage`, então o evento contém uma nota. Como o algoritmo parte do último evento para o primeiro, a última mensagem que deverá ser encontrada é a de `NOTE_OFF` (linha 01). O Quadro 6 mostra o trecho de código em que isso é

feito.

```
01 if (sm.getCommand() == Constants.NOTE_OFF || sm.getData2() == 0) {
02   key = sm.getData1();
03   octave = (key / 12) - 1;
04   note = key % 12;
05
06   noteList.put(scale.getNote(note).toString() + octave,
07   midiEvent.getTick());
08 }
```

Quadro 6 – Identificação das notas

O tom é obtido na mensagem chamando o método `getData1` e o resultado armazenado na variável `key` (linha 02). É necessário ainda identificar a oitava, dividindo `key` por 12 e subtraindo um. Esse ajuste se deve aos semitons que formam a escala cromática. A nota por fim é obtida, efetuando a uma operação de resto de uma divisão por 12. A seguir a informação da nota é inserida em uma lista auxiliar, juntamente com a marcação de tempo em que o evento ocorreu, sendo obtido pelo método `getTick` (linha 07).

A próxima mensagem enviada pelo arquivo MIDI deve ser para começar a tocar uma nota, ou seja, `NOTE_ON`. O Quadro 7 exibe o trecho do algoritmo que executa esta operação.

```
01 if (sm.getCommand() == Constants.NOTE_ON && sm.getData2() > 0) {
02   key = sm.getData1();
03   octave = (key / 12) - 1;
04   note = key % 12;
05
06   if (!noteList.containsKey(scale.getNote(note).toString() + octave))
07       return;
08
09   long tickDuration = noteList.remove(scale.getNote(note).toString()
10                       + octave) - midiEvent.getTick();
11
12   ENoteValue noteValue = getNoteValue(tickDuration);
```

Quadro 7 – Identificação de figuras rítmicas

Ao identificar a nota, a mesma é procurada na lista auxiliar (linha 09). Se ela for encontrada, a lista retornará a sua duração. A duração da nota e por consequência, sua figura rítmica, é definida subtraindo-se o tempo final do atual.

A figura rítmica é definida utilizando o método `getNoteValue`. Neste método divide-se a diferença entre a marcação de tempo final e a atual da nota, representada pela variável `tickDuration`, pelo valor da semínima. O método é mostrado no Quadro 8.

```
01 private ENoteValue getNoteValue(long ticks) {
02             float value = (float) ticks / quarterNoteValue;
03
04
05             if ( value == 4)
06                   return ENoteValue.WHOLE_NOTE;
07
08             if ( value == 2)
09                   return ENoteValue.HALF_NOTE;
10
11             if ( value == 1)
12                   return ENoteValue.QUARTER_NOTE;
13
14             if (value == 0.5)
15                   return ENoteValue.EIGHTH_NOTE;
16
17             if (value == 0.25)
18                   return ENoteValue.SIXTEENTH_NOTE;
19
20             if (value == 0.125)
21                   return ENoteValue.THIRTY_SECOND_NOTE;
22
23             if (value == 0.0625)
24                   return ENoteValue.SIXTY_FOURTH_NOTE;
25
26             return null;
27       }
```

Quadro 8 – Retornando o nome das figuras rítmicas

A semínima foi escolhida por existir um método da classe `Sequence` que retorna seu valor no arquivo MIDI, `getResolution`. A partir desse valor é possível encontrar todas as outras figuras rítmicas. O valor encontrado é então comparado com valores previamente definidos.

Quando não é possível encontrar uma figura rítmica para o valor obtido, o retorno do método é nulo. Neste caso, primeiramente tenta-se verificar se a nota não possui pontos de aumento. Isto é feito no método `useDottedNotes`, que consiste em encontrar uma figura rítmica para o valor passado e utilizar o restante do valor para encontrar outra figura rítmica, que necessariamente deve ser a metade da anterior. O Quadro 9 exibe o algoritmo.

```
01 private int useDottedNote(long tickDuration) {
02       int count = 0;
03       long duration = tickDuration;
04       ENoteValue mainNoteValue = getMainNoteValue(tickDuration);
05       duration-= noteValueToInt(mainNoteValue);
06       while (count < 3 && duration > 24){
07                   mainNoteValue = getMainNoteValue(duration);
08             duration -= noteValueToInt(mainNoteValue);
09             count++;
10       }
11       return count;
12 }
```

Quadro 9 – Verificação de notas com ponto de aumento

Primeiramente, o algoritmo subtrai do tempo informado uma figura rítmica principal, que deve ser a maior possível, utilizando método `getMainNoteValue` (linha 04). Em seguida, com o restante do tempo, procura-se outras figuras rítmicas menores, que serão sempre divisões da figura principal (linha 08). Em seguida, a variável `count` é incrementada. Este processo é repetido até que a condição de entrada não seja mais aceita. Por fim, a variável `count` é retornada, indicando quantos pontos de aumentos tem a nota.

O retorno do método indica quantos pontos de aumento serão utilizados, sendo que não poderá ser maior do que dois, pois é o limite das partituras de tinta. Se não for possível identificar a duração da nota, parte-se para o uso de ligaduras de duração, utilizando o método `getTie`.

Para o processamento das ligaduras, a lógica é similar à utilizada no método `getDottedNotes`. Define-se a duração da nota principal e em seguida divide-se o tempo restante até que este seja zero. O Quadro 10 mostra o algoritmo.

```
01 private Tie getTie(int note, int octave, long tickDuration) {
02       Tie aTie = new Tie();
03       ENoteValue mainNoteValue = getMainNoteValue(tickDuration);
04
05       aTie.addNote(new Note(scale.getNote(note), octave,
06              mainNoteValue, null));
07
08       tickDuration = tickDuration - noteValueToInt(mainNoteValue);
09       do {
10             mainNoteValue = getMainNoteValue(tickDuration);
11
12             aTie.addNote(new Note(scale.getNote(note), octave,
13                   mainNoteValue, null));
14
15             tickDuration = tickDuration - oteValueToInt(mainNoteValue);
16       } while (tickDuration > 0);
17
18       return aTie;}
```

Quadro 10 – Ligadura de duração

Primeiro é necessário extrair do tempo informado a maior figura rítmica possível (linha 03). Em seguida, esta nota é inserida em uma lista de notas que compõem a ligadura (linha05). Como se trata de uma ligadura de duração, as notas são iguais à primeira nota inserida nesta lista. Com o tempo restante, são extraídas tantas figuras rítmicas quanto forem possíveis, adicionando-as à lista (linha 12). O retorno do método é um objeto `Tie`.

O elemento de pausa é processado de forma semelhante ao das notas, exceto pelo fato de que não há ligaduras de duração entre as pausas, nem estas possuem tons. O Quadro 11 detalha este algoritmo.

```
01 public void update(IObservable aObservable, MidiEvent midiEvent) {
02       long restTime = 0l;
03
04       if (aObservable == noteProcessing) {
05             noteProcessing = (MidiNoteProcessing) aObservable;
06             MidiMessage message = midiEvent.getMessage();
07             if (message instanceof ShortMessage) {
08                   ShortMessage sm = (ShortMessage) message;
09                   if (sm.getCommand() == Constants.NOTE_ON) {
10                         // adiciona o tempo para calcular a pausa
11                         pauseList.put("pausa", midiEvent.getTick());
12                   } else if (sm.getCommand() == Constants.NOTE_OFF) {
13                         if (pauseList.get("pausa") != null)
14       restTime = (pauseList.remove("pausa") - midiEvent.getTick());
15       restTime = restTime - (restTime / 10);
16
17       if (restTime > 0) {
18          ERestValue aPause = getRestValue(restTime-(restTime / 10));
19
20       if (aPause != null) {
21          noteProcessing.getElementList().addElement(new Rest(aPause));
22       } else {
23          //procura pausa com ponto de aumento
24         if (aDot != null) {
25            ERestValue anotherPause = getMainRestValue(restTime);
26            noteProcessing.getElementList().addElement(
27                                              new Rest(anotherPause, aDot));
28             }
29
30           }
31
32         }
33
34       }
35
36    }
37 }
```

Quadro 11 – Identificação das pausas

Para encontrar a duração da pausa, efetua-se o cálculo com base na diferença entre os eventos de `NOTE_ON` e `NOTE_OFF` (linha 18), pois este é o tempo entre uma nota e outra. Da mesma forma que ocorre com o processamento das notas, utiliza-se uma lista auxiliar para que seja possível calcular a duração da pausa. Se este tempo for tão pequeno que não se encaixa em nenhuma figura de pausa, a pausa é desconsiderada. Caso contrário, um elemento de pausa é criado.

Da mesma forma que ocorre com as notas, caso não seja possível identificar a duração da pausa, o método `useDottedNote` (linha 23) é chamado. O elemento de pausa é inserido, por fim, na lista de elementos da música (linha 26).

O processamento dos compassos ao mesmo tempo em que os elementos de nota e pausa vão sendo inseridos na lista. Para identificar onde inicia e termina cada compasso, a cada vez que o método `update` executa, é feita a contagem da duração dos elementos

inseridos até então, utilizando para isso o método `isBarFull`, detalhado no Quadro 12.

```
01 private boolean isBarFull() {
02             int count = 0;
03             ENoteValue aNoteValue;
04             ERestValue aRestValue;
05             for (IElement e : elementList) {
06                   if (e instanceof Note) {
07                         aNoteValue = ((Note) e).getNoteValue();
08
09                         if (((Note) e).getDotType() != null) {
10                               switch (((Note) e).getDotType())
11                         // processamento da duração das notas
12                         } else {
13                           if (e instanceof Rest) {
14                            // processamento da duração das pausas
15                            } else {
16                              count += getRestValue(aRestValue);
17                            }
18                       }
19                   }
20               }
21             return count == valuePerBar;
22 }
```

Quadro 12 – Identificação dos elementos de compasso

No momento da criação das classes observadas, é definido o tempo total de cada compasso, valor este armazenado pelo atributo `valuePerBar`. Cada elemento inserido na lista é lido e seu valor é acumulado. O método retorna se o compasso está completo (linha 21).

Após a contagem ter sido feita, o valor do somatório é comparado com o definido anteriormente. Caso o valor seja o total de um compasso, um elemento `Bar` é inserido na lista de elementos. As próximas contagens serão feitas a partir deste último compasso.

Além do cálculo de compasso, a classe `MidiBarProcessing` é responsável por definir a clave e o tom em que a música está escrita. O Quadro 13 mostra o trecho do método que retorna a armadura da clave e o método `getClef`.

```
01 private Clef getClef(MetaMessage aMessage) {
02       switch (aMessage.getMessage()[1]) {
03         case 0x58:
04           int count = aMessage.getData()[0];
05           int noteValue = 1 << aMessage.getData()[1];
06           if (clef == null) {
07             clef = new Clef(ENote.C, count,
08                             ENoteValue.valueOf(noteValue),null);
09             valuePerBar = count *
10                           getNoteValue(ENoteValue.valueOf(noteValue));
11             } else {
12               clef.setBeat(count);
13               clef.setNoteValue(ENoteValue.valueOf(noteValue));
14               valuePerBar = count *
15                           getNoteValue(ENoteValue.valueOf(noteValue));
16             }
17             break;
18       case 0x59:
19             if (clef == null)
20             clef = new Clef(getKey(aMessage.getMessage()[3]), 4,
21                             ENoteValue.valueOf(4), null);
22             else {
23                   clef.setKey(getKey(aMessage.getMessage()[3]));
24                  }
25             break;
26       }
27       return clef;}
```

Quadro 13 - Identificação do tom da música

Nos arquivos MIDI utiliza-se a faixa zero para as informações de tom. Por este motivo, é necessário efetuar o processamento da faixa zero antes de processar a faixa escolhida. Estas informações são enviadas em objetos do tipo `MetaMessage`, acessadas pelo método `getMessage` (linha 02). Este método retorna um vetor que indica as informações acerca do tom e da fórmula de compasso. A mensagem pode retornar a fórmula de compasso, utilizando a posição zero do vetor para a quantidade de figuras e a posição 1 para a figura rítmica utilizada (linha 05). O tom é definido nesse objeto por um numero que varia de -7 a 7, indicando a quantidade de bemóis, de -7 a -1, ou sustenidos, de 1 a 7 e o nome vem da enumeração `ENote`. Caso o atributo retorne zero, o tom é dó maior, ou lá menor (linha 22).

Após processar os elementos do arquivo MIDI, é feita a conversão do seu conteúdo para o sistema de musicografia braile. A lista de elementos da música é a entrada do método `metodoConverteBraile`, responsável pela conversão, conforme mostrado pelo Quadro 14.

```
01 public void processElements() {
02   IElement aElement = null;
03   do {
04     aElement = elementList.getNextElement();
05     if (aElement instanceof Note) {
06        braileNoteList.addElement(processNoteElements(aElement));
07        lastNote = (Note) aElement;
08     } else {
09        if (aElement instanceof Rest) {
10           braileNoteList.addElement(processRestElements(aElement));
11        } else {
12         if (aElement instanceof Bar) {
13            braileNoteList.addElement(BraileConstants.BAR);
14         } else {
15           if (aElement instanceof Clef) {
16              braileNoteList.addElement(processClefElement(aElement));
17           } else {
18             if (aElement instanceof Tie) {
19                braileNoteList.addElement(processTieElement(aElement));
20           }
21         }
22        }
23      }
24    }
25  } while (aElement != null);
26 }
```

Quadro 14 - Conversão para o sistema de musicografia braile

Os métodos de processamento estão separados de acordo com o tipo de elemento, porém todos os métodos apenas traduzem os símbolos para o código musicográfico braile, não havendo nenhum processamento especial neles. O método insere os elementos convertidos em uma lista de strings que será exibida na área de texto da ferramenta. Esta área de texto está configurada para trabalhar com uma fonte braile, exibindo dessa forma a partitura convertida.

3.3.2 Operacionalidade da ferramenta

Os passos para gerar e imprimir uma partitura braile são demonstrados nesta seção. A interface da ferramenta é simples, sendo formada por cinco botões e uma área de texto onde será apresentada a partitura braile gerada. É importante ressaltar que a utilização da ferramenta destina-se a pessoas videntes e por este motivo, não foram implementados quaisquer recursos de assistência por voz ou outro tipo de ferramenta de acessibilidade. A Figura 25 mostra a tela principal da ferramenta.

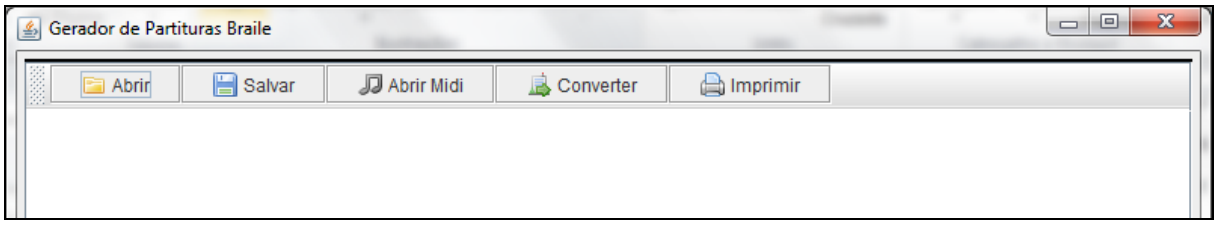


Figura 25 – Tela principal da ferramenta

O primeiro passo para a geração de uma partitura braile é carregar um arquivo MIDI previamente gerado em algum software editor de partituras, como o Encore, por exemplo. Isto é feito clicando no botão `Abrir Midi`. A ferramenta somente permite a geração de uma partitura braile a cada vez, e por esse motivo, é necessário selecionar a faixa MIDI para a qual se deseja efetuar a geração e clicar em OK, conforme mostrado pela Figura 26.

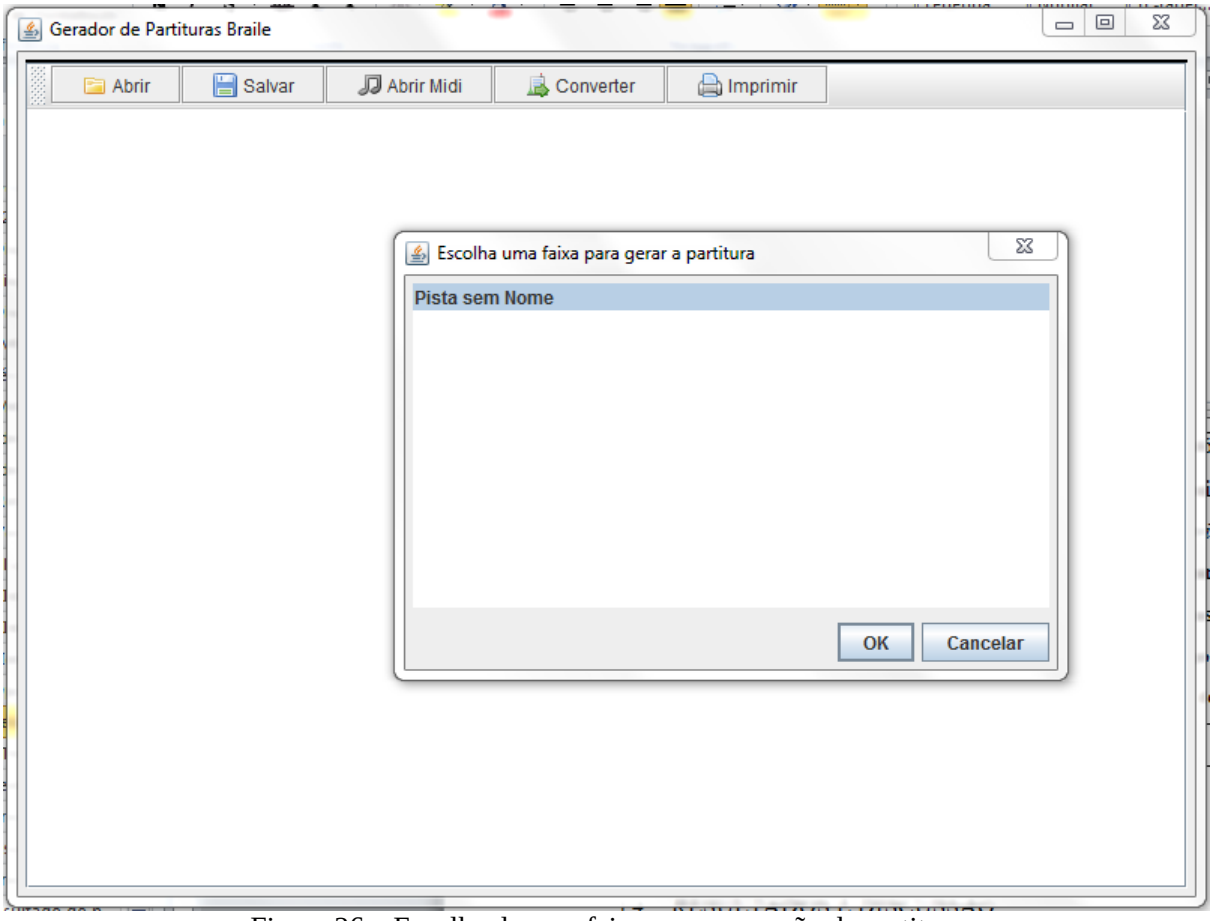


Figura 26 – Escolha de uma faixa para a geração da partitura

Quando a faixa do arquivo MIDI não tem um nome, um nome padrão é apresentado. Devido ao propósito educacional desta ferramenta, a geração de uma partitura de cada vez não chega a ser uma limitação.

Após a seleção da faixa, basta clicar no botão *Converter* para iniciar o processamento do arquivo MIDI e a geração da partitura, que é apresentada na tela, conforme mostra a Figura 27.

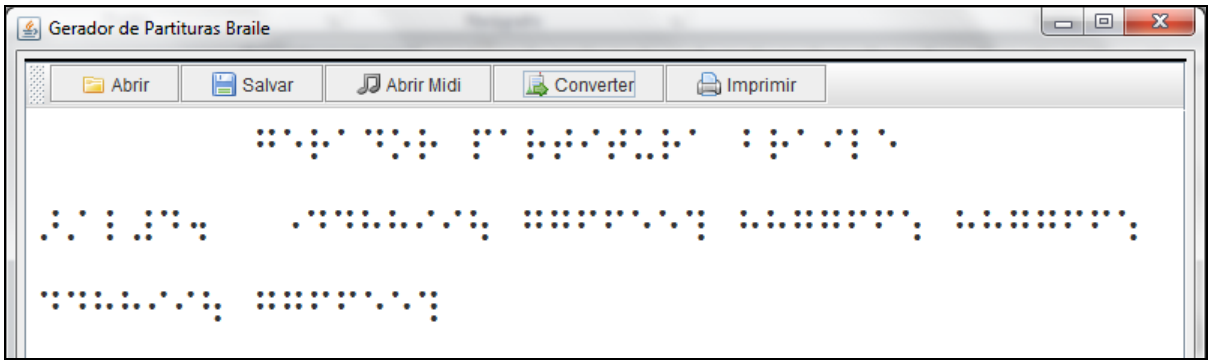


Figura 27 – Partitura braile gerada e mostrada na tela

É possível salvar a partitura gerada para ser impressa posteriormente. Para isso, deve-se clicar no botão *Salvar*. Para abrir uma partitura previamente gerada, deve-se clicar no botão *Abrir*. Por fim, para imprimir a partitura gerada, basta clicar no botão *Imprimir*. Deve-se, entretanto, ter uma impressora braile instalada e configurada. Uma impressora braile é um tipo de impressora de impacto capaz de imprimir células táteis no papel. Essa impressora funciona de forma similar às impressoras matriciais, com a exceção de que não há fita de impressão. A cabeça de impressão é prensada contra o papel, imprimindo assim, os caracteres braile. Da mesma forma que as impressoras comuns, o resto do trabalho é realizado pelo *driver* do dispositivo.

Com a ferramenta desenvolvida é possível gerar diversas partituras utilizando diversos símbolos musicais a partir de arquivos MIDI, possibilitando a inclusão de alunos com deficiência visual em cursos de música. Para comprovar o funcionamento a ferramenta, foram executados diversos testes que estão descritos na seção seguinte.

3.4 RESULTADOS E DISCUSSÃO

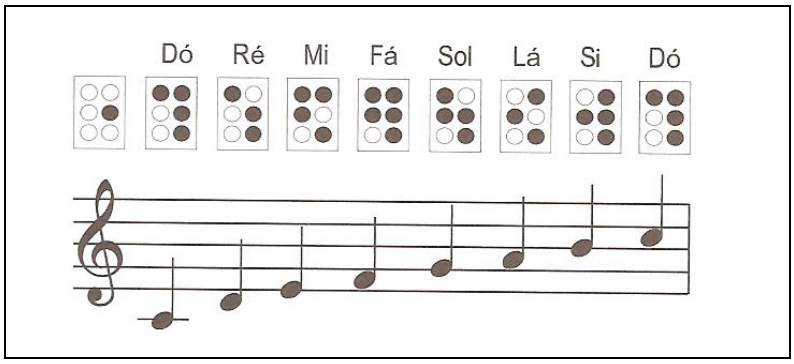
Nesta seção são descritos os testes realizados para comprovar o funcionamento da ferramenta desenvolvida e verificar a acurácia da partitura braile gerada.

Esta seção foi dividida em duas subseções, na seção 3.4.1 é descrito como foram feitos os testes com trechos musicais abrangendo diversos aspectos da musicografia braile e também são discutidos os resultados obtidos. Na seção 3.4.2 é descrito como foram feitos os testes com músicas completas, comparando os resultados obtidos com a partitura gerada pelo software Musibraille.

3.4.1 Testes realizados com trechos musicais

O objetivo deste teste é validar pequenos trechos musicais isolados, cada um testando um aspecto da musicografia braile. Os exemplos mesclam figuras rítmicas diferentes, pausas, sinais de acidente, comparando-os com Tomé (2003). Alguns exemplos estão incompletos nesse trabalho, destacando a parte relevante à seção e ignorando a completude dos compassos. Nesses casos, são adicionadas notas ao arquivo MIDI utilizado no teste para completar o compasso corretamente. O resultado obtido está destacado nas imagens desta seção. Convém lembrar que a ferramenta gera as partituras com a armadura de clave e fórmula de compasso. Também é escrito o título da música na primeira linha da partitura, ou o nome da ferramenta, na falta do título. Além disso, algumas notas são precedidas por um ponto, indicando a oitava. Os exemplos foram escritos utilizando o software Mozart versão 11.

Para o primeiro exemplo, a Figura 28 mostra uma escala de dó maior formada por semínimas e sua notação em braile.



Fonte: Tomé (2003, p. 38).

Figura 28 – Escala de dó maior com semínimas e sua representação braile

O exemplo foi escrito para MIDI no andamento 4/4 e o resultado obtido pela ferramenta para este teste é mostrado na Figura 29.

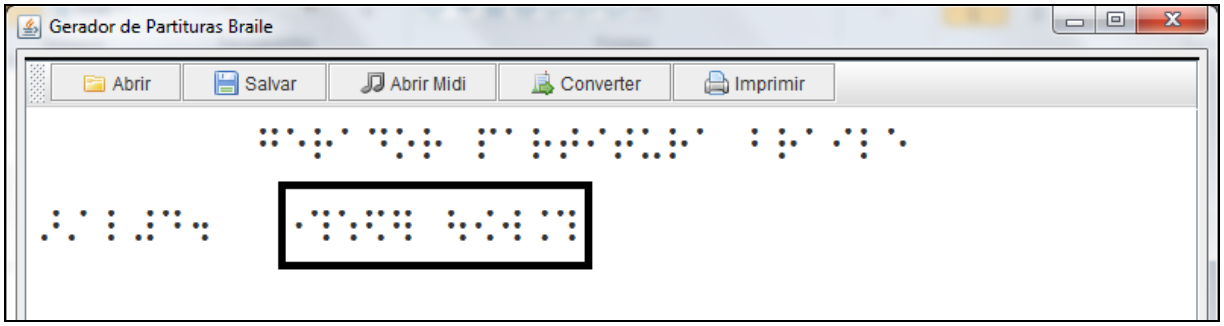
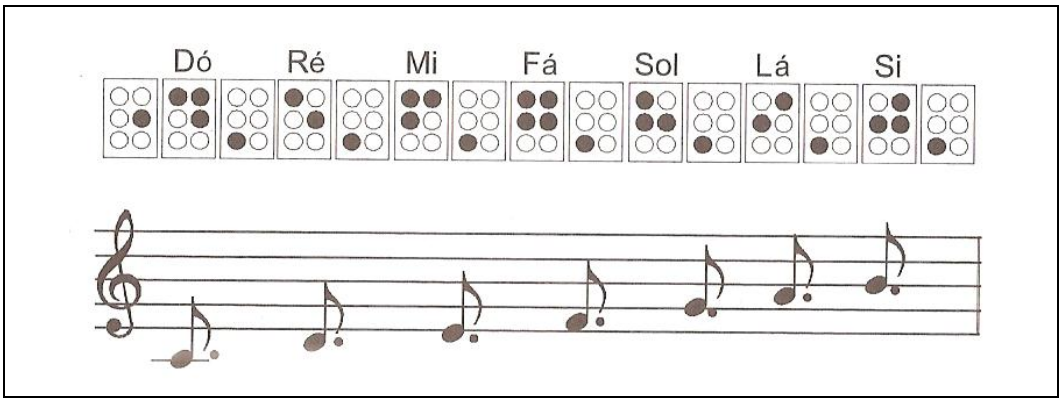


Figura 29 – Resultado obtido para a interpretação de semínimas

Houve uma única diferença entre a partitura gerada pela ferramenta e o apresentado no livro. Trata-se do símbolo da quinta oitava que precede a última nota. Este símbolo é

necessário para que não haja confusão na leitura da música e o dó da quarta oitava seja tocado. Exceto por isso, a partitura gerada pela ferramenta é igual à indicada por Tomé (2003).

Para o teste de notas com ponto de aumento, a Figura 30 apresenta uma escala de dó maior, escrita em colcheias com ponto de aumento.



Fonte: Tomé (2003, p. 46).

Figura 30 – Colcheias com ponto de aumento e sua notação braile

Todas essas notas juntas ultrapassam a medida de um compasso 4/4. Desta forma, para adequar o teste, foi escrito um MIDI nas mesmas condições, porém com a nota lá em semicolcheia. Uma colcheia aumentada equivale a três semicolcheias. Sendo a semicolcheia a quarta parte de uma semínima, em um compasso 4/4 são necessárias dezesseis semicolcheias. Cinco colcheias aumentadas equivalem a quinze semicolcheias, sendo necessária a semicolcheia de lá para completar o compasso. O resultado obtido pela ferramenta é exibido pela Figura 31.

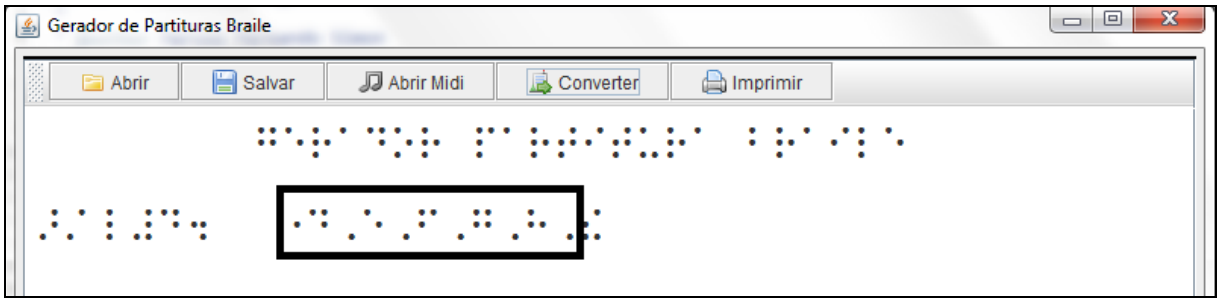
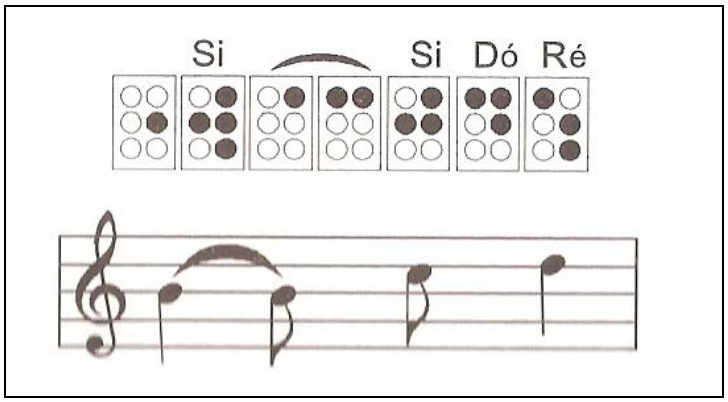


Figura 31 – Resultado do processamento de colcheias com ponto de aumento

Analisando as figuras acima se pode perceber que a geração da partitura pela ferramenta deu-se de forma correta, apresentando o mesmo resultado indicado por Tomé (2003).

Para o teste de ligadura de duração, a Figura 32 apresenta uma pequena frase musical.



Fonte: Tomé (2003, p. 45).

Figura 32 – Ligaduras de duração

O resultado obtido gerando a partitura braile na ferramenta é exibido na Figura 33.

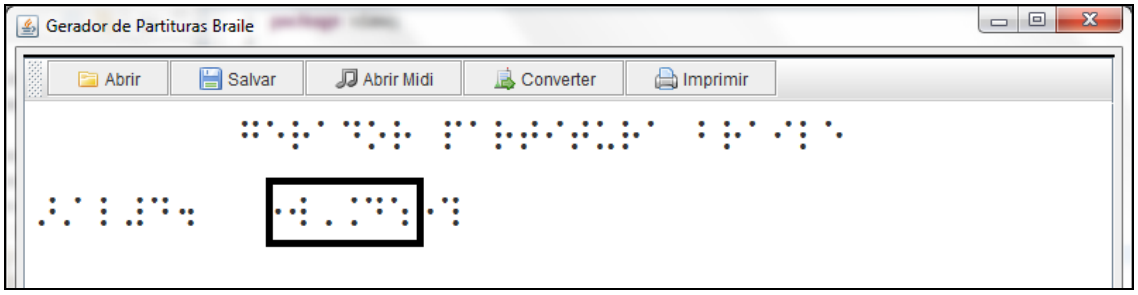
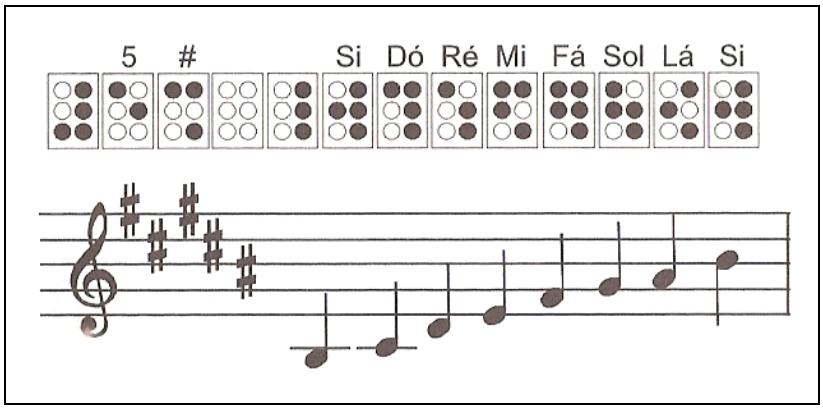


Figura 33 – Ligadura de duração e seu equivalente com ponto de aumento

É possível notar que a partitura gerada pela ferramenta interpretou a duração da nota como se houvesse um ponto de aumento e não uma ligadura de duração. Neste caso, o resultado divergiu por causa de uma regra adotada na ferramenta. A duração da nota somente será representada utilizando a ligadura de duração caso não seja possível sua representação com pontos de aumento. Na prática a leitura é equivalente, pois a duração da nota é a mesma. Outro ponto que divergiu da partitura apresentada no exemplo foi o símbolo da quinta oitava antes da nota dó em semicolcheia.

O exemplo seguinte apresenta uma escala de si maior em semínimas, com o objetivo de demonstrar a armadura de clave com mais de três sustenidos. A notação é mostrada na Figura 34.



Fonte: Tomé (2003, p. 72).

Figura 34 – Escala de si maior e a armadura de clave com cinco sustenidos

A ferramenta gera a partitura conforme mostra a Figura 35.

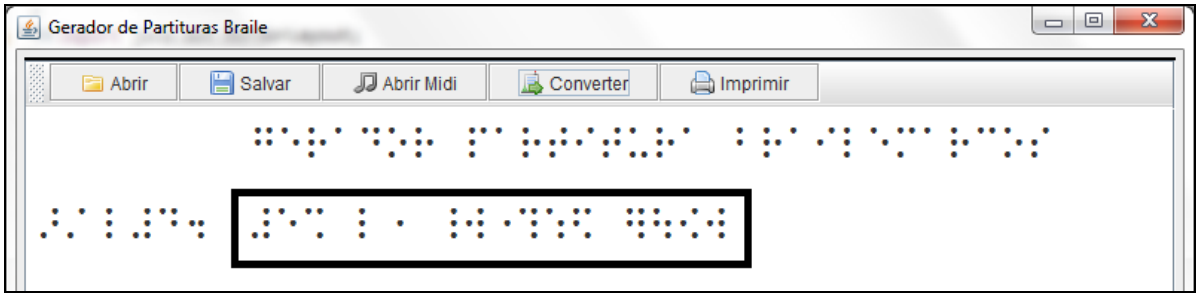
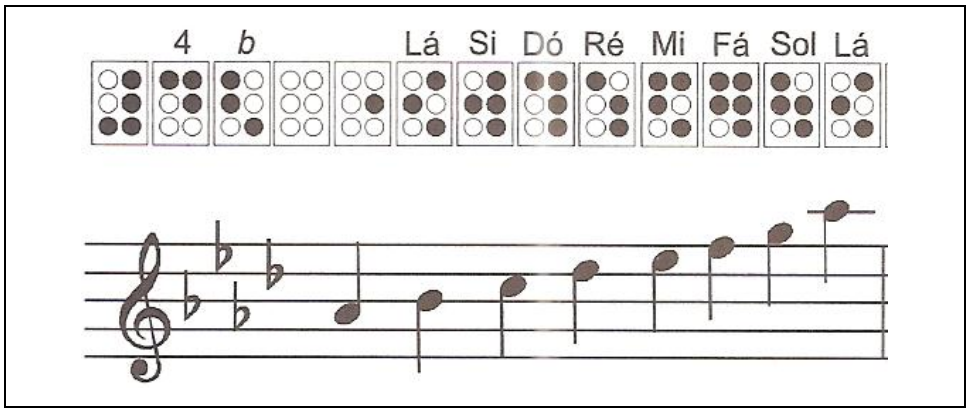


Figura 35 – Armadura de clave com sustenidos gerada pelo programa

Analisando as figuras acima é possível verificar que a armadura de compassos foi escrita pela ferramenta da mesma forma que indicada por Tomé (2003), exceto pelo emprego das oitavas, conforme indicado por Tomé (2003, p.57).

A Figura 36 mostra uma escala de lá bemol escrita em semínimas, com o objetivo de mostrar a armadura de claves com bemóis.



Fonte: Tomé (2003, p. 74).

Figura 36 – Escala de lá bemol e armadura de clave com 4 bemóis

A ferramenta gera a partitura conforme mostra a Figura 37.

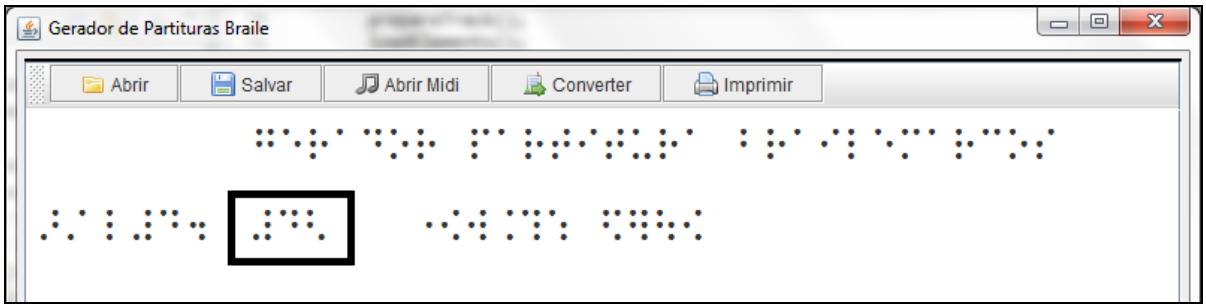
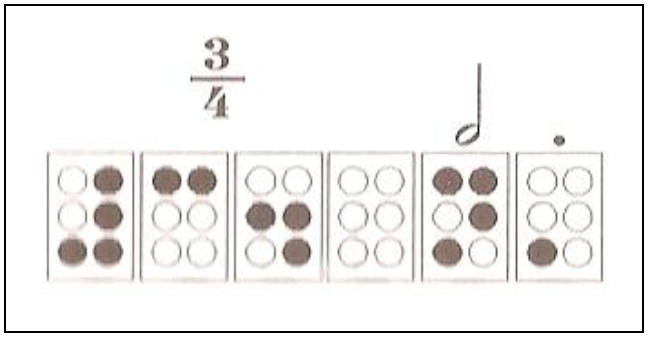


Figura 37 – Armadura de clave com bemóis gerada pelo programa

Conforme mostra a imagem acima, a ferramenta gerou a partitura como a indicada por Tomé (2003). A ferramenta apresenta os símbolos das oitavas, necessários para a execução correta do trecho musical.

O exemplo seguinte, na Figura 38 mostra um trecho musical escrito em outra fórmula de compasso: ¾.



Fonte: Tomé (2003, p. 51).

Figura 38 – Fórmula de compasso em ¾

A Figura 39 mostra a partitura gerada pela ferramenta.

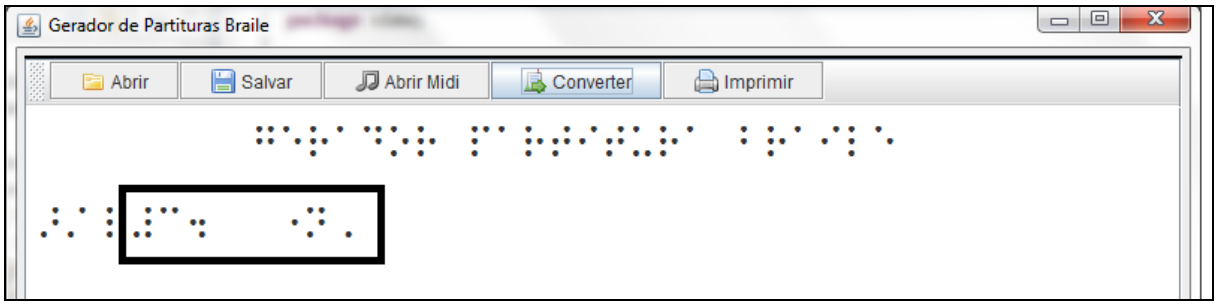


Figura 39 – Partitura com fórmula de compasso em 3/4

A parte fora do destaque é referente à clave de sol. Os três primeiros pontos da área destacada referem-se à fórmula de compasso ¾, conforme indicado por Tomé (2003). O ponto que aparece antes da nota é referente à quarta oitava, a qual a nota pertence. É possível, portanto, verificar que a ferramenta gerou a partitura corretamente.

Os testes executados nesta subseção representam elementos comuns à maioria das partituras. Foi verificado que a ferramenta foi capaz de reconhecer a duração das notas e gerar corretamente as partituras no sistema musicográfico braile. Entretanto, Tomé (2003) não

apresenta exemplos de notas e pausas em um mesmo trecho, de modo que essa situação não pôde ser verificada. A próxima subseção verifica a geração de partituras com trechos musicais maiores e que formam melodias completas, não apenas escalas ascendentes, fazendo, eventualmente o uso de figuras pausa e conferindo assim o funcionamento dessa propriedade.

Com relação aos objetivos deste trabalho, pode-se afirmar que esses foram atingidos e que este trabalho permite a geração de partituras braile, possibilitando assim a inclusão de alunos com deficiência visual em cursos de música.

3.4.2 Testes realizados em ambiente real

Esta seção demonstra os testes realizados com musicais reais. Foram escolhidas duas músicas para validar a geração da partitura braile realizada pela ferramenta. Para os testes foi utilizado o software Musibraille, pois este permite baixar diversas partituras em braile e gerar o arquivo MIDI para elas, permitindo assim comparar os resultados obtidos pela ferramenta. Entretanto, uma dificuldade encontrada utilizando o Musibraille foi a geração de arquivos MIDI em tons diferentes de dó maior. O programa escreve as notas no arquivo conforme estas são tocadas, não enviando uma mensagem que identifica o tom. Também houve dificuldades para gerar partituras em outras fórmulas de compasso. Em ambos os casos, para permitir um conjunto de testes mais amplo, optou-se por escrever a partitura e gerar o arquivo MIDI no editor Mozart 11.

O primeiro teste foi feito com um trecho da melodia de Ode à Alegria, de Ludwig van Beethoven, que está escrita na fórmula de compasso 4/4. A Figura 40 mostra a partitura escrita no programa Musibraille.

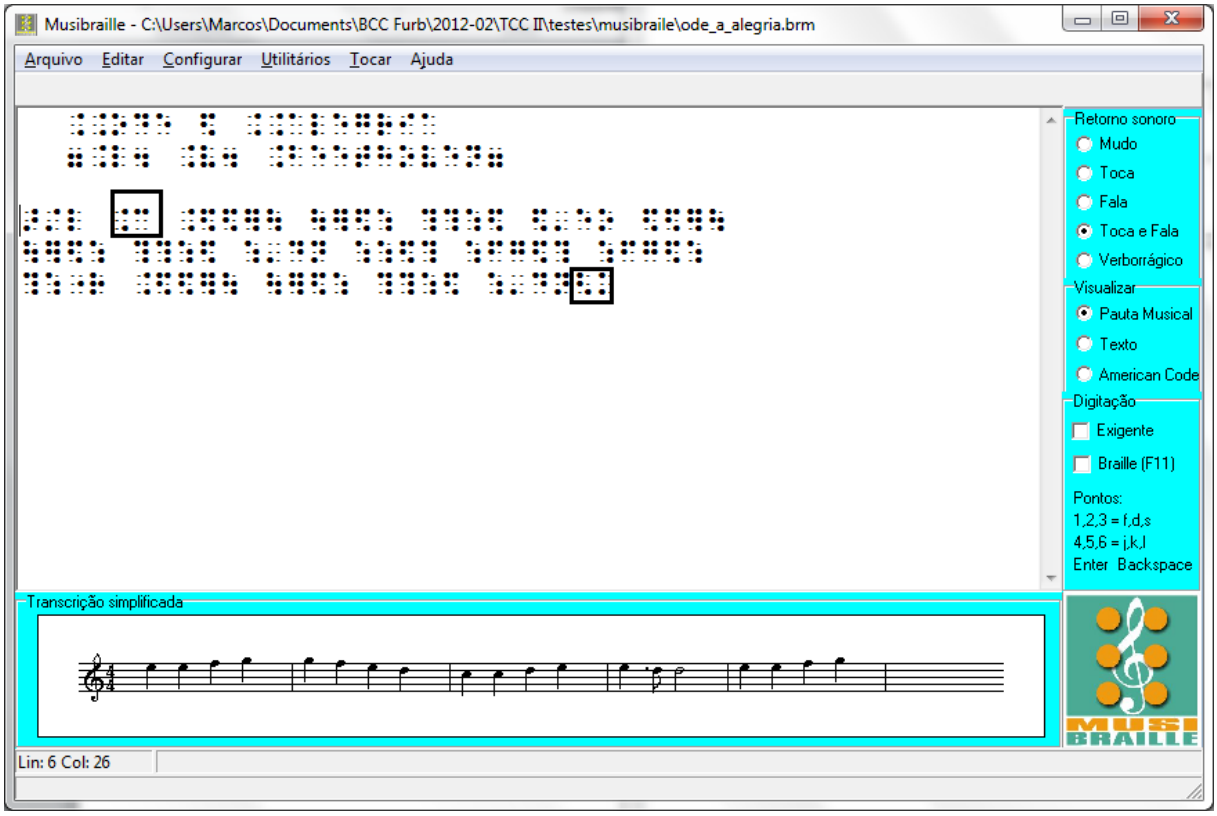


Figura 40 – Partitura escrita no Musibraille

A partir do arquivo MIDI gerado pelo programa, foi gerada a partitura na ferramenta. Os resultados obtidos são apresentados na Figura 41.

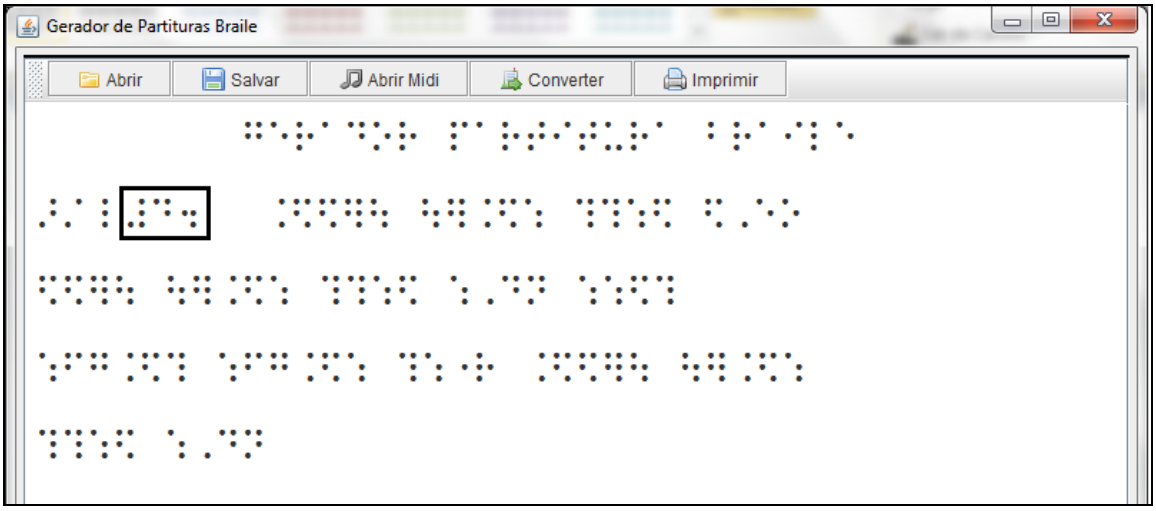


Figura 41 – A mesma partitura, gerada na ferramenta

Os resultados alcançados divergem em alguns pontos como, por exemplo, a identificação das oitavas antes de algumas notas. No inicio da partitura, os três primeiros símbolos referem-se à clave de sol. Logo após, a ferramenta escreve a fórmula de compasso como 4/4, o que está correto. No entanto, conforme indica o primeiro destaque da Figura 40, o software Musibraille escreve essa fórmula de uma maneira alternativa, utilizando um símbolo formado por dois caracteres com os pontos 4 e 6 e 1 e 4. O símbolo final da partitura gerada

pelo Musibraille, indicado no segundo destaque, é formado por dois caracteres com os pontos 1, 2 e 6 e 1 e 3, refere-se à barra de repetição de compasso, símbolo não presente no escopo desta ferramenta O Quadro 15 mostra os resultados obtidos neste teste.

Símbolos totais	Símbolos processados corretamente	Símbolos divergentes	Símbolos não processados
73	68	4	1
Percentual	93,15%	5,47%	1,38%

Quadro 15 – Resultado do teste com a música Ode à Alegria

Observando o Quadro 15 é possível notar que 5,47% dos símbolos da partitura foram processados de forma divergente da partitura gerada no Musibraille, sendo esses símbolos referentes a oitavas. O símbolo que não foi processado, conforme mencionado acima é a barra de retorno e não faz parte do escopo deste trabalho.

O segundo teste foi realizado com a cantiga de roda O Cravo Brigou com a Rosa. Esta música está escrita na fórmula de compasso ¾. No arquivo utilizado no teste, além da partitura braile para música, está contido o texto com a letra. Como isso está fora do escopo deste trabalho, destacou-se na Figura 42 a transcrição da música.

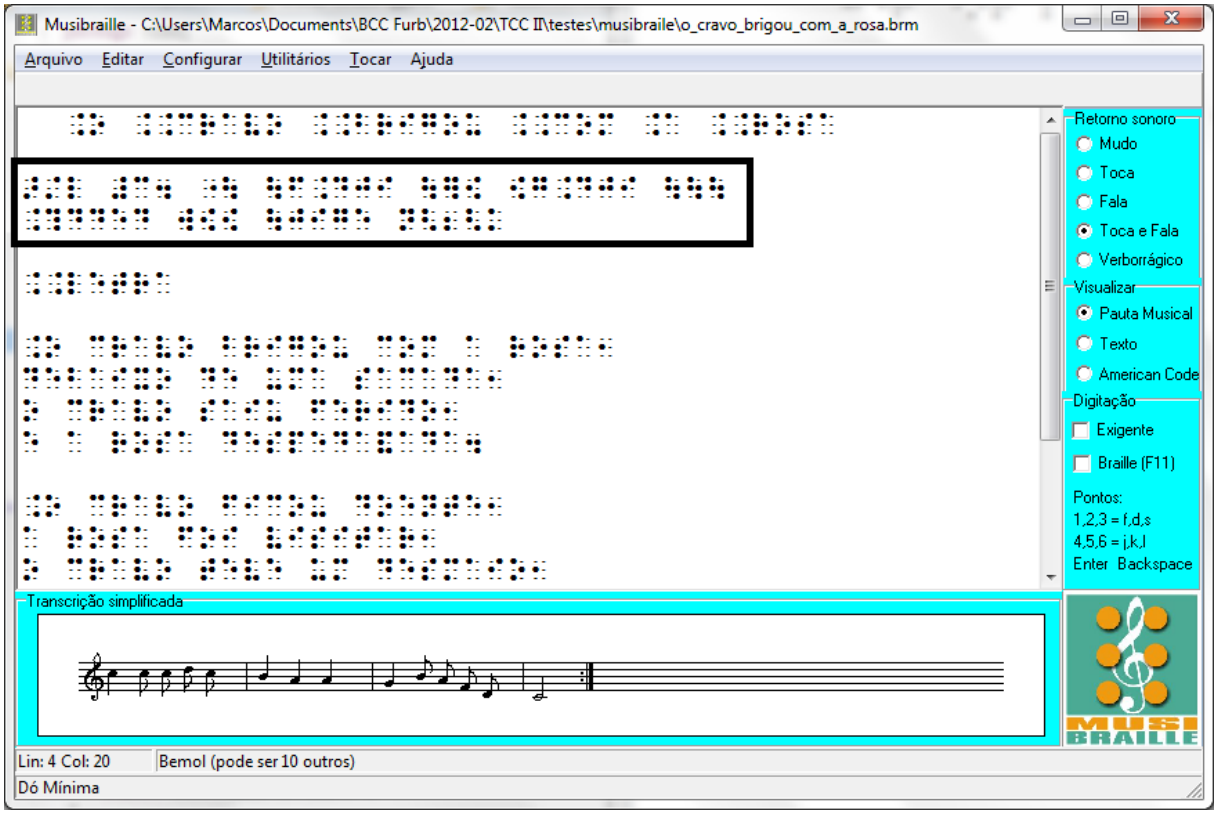


Figura 42 – Partitura gerada pelo Musibraille

A partitura gerada pela ferramenta é mostrada na Figura 43.

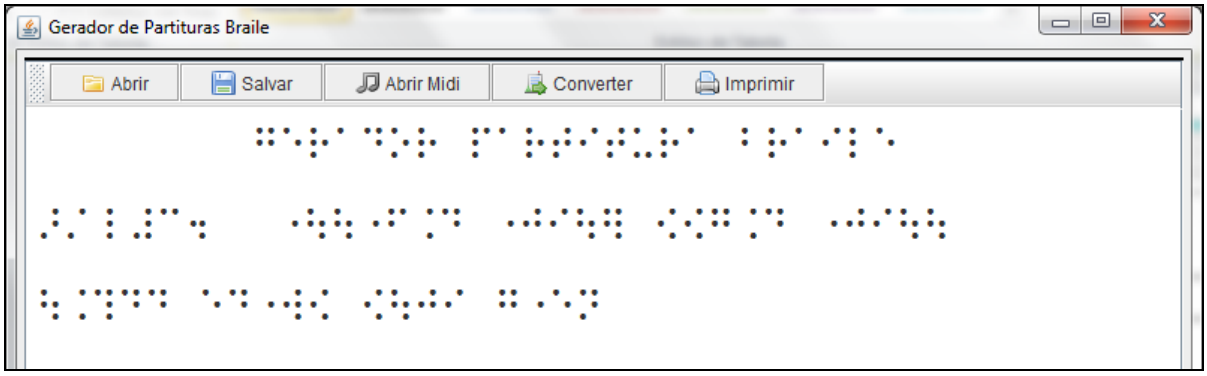


Figura 43– Partitura gerada pela ferramenta

A partitura gerada pela ferramenta difere ligeiramente da gerada pelo Musibraille em dois pontos: o uso de anacruse e a barra de repetição. A anacruse é a ausência de tempos no primeiro compasso de uma música. Isso significa que a música não começa no primeiro tempo do compasso e estas notas faltantes são completadas ao final da música. No caso da música utilizada no teste, a primeira nota sol é tocada no terceiro tempo do compasso. No entanto, devido a problemas na geração do arquivo MIDI, optou-se por escrever a música em um editor de partituras e gerar o arquivo MIDI para permitir a geração da partitura braile. Neste caso não houve diferenças na execução da música por causa da anacruse porque foi possível escrever a música de modo a não utilizar esta notação. Todavia, há casos que isso não é possível, e nesses casos, a partitura será gerada incorretamente. O Quadro 16 detalha os resultados obtidos.

Símbolos totais	Símbolos processados corretamente	Símbolos processados incorretamente	Símbolos não processados
45	39	4	2
Percentual	86,66%	8,89%	4,45%

Quadro 16 – Resultados do teste com a música O Cravo Brigou Com a Rosa

Uma análise do Quadro 16 nos permite verificar que a partitura foi gerada de forma aceitável, ainda que divirja em alguns pontos. O uso da anacruse foi desconsiderado, de modo que as diferenças entre as duas partituras se deveu apenas aos símbolos que indicam a oitava, conforme mencionado acima. Os símbolos não processados pela ferramenta são barras de repetição e não estão no escopo deste trabalho.

Os testes descritos acima mostraram que a ferramenta é capaz de gerar partituras em braile para músicas simples. Apesar de algumas notações serem escritas de outra maneira na ferramenta, o significado se mantém e a interpretação da peça musical é o mesmo.

Alguns aspectos da musicografia braile não foram abordados e algumas situações não foram tratadas. Em se tratando de uma primeira versão e do tempo gasto no desenvolvimento, é possível relevar estas limitações. Alguns desses aspectos são a utilização de quiálteras,

anacruses e acordes. Essas limitações estão mais bem detalhadas na seção sobre as limitações da ferramenta. A seguir, as conclusões finais discorrem sobre os resultados obtidos.

4 CONCLUSÕES

O presente trabalho propôs o desenvolvimento de uma ferramenta que interpretasse arquivos MIDI e os convertesse para o sistema de musicografia braile. Este trabalho propõe uma alternativa acessível a softwares comerciais que realizam essa função.

Para o desenvolvimento da ferramenta, foi utilizada a linguagem de programação Java, em sua versão 1.7. Para a leitura e interpretação dos arquivos MIDI foi utilizada a biblioteca Java Sound. A partir dos elementos do arquivo MIDI devidamente interpretados e classificados foi então possível efetuar a tradução para o sistema musicográfico braile, conforme indicado por Tomé (2003). Devido ao grande número de elementos musicográficos e possibilidades na escrita musical, a ferramenta foi desenvolvido visando interpretar os símbolos mais comumente utilizados na escrita musical.

Os resultados demonstram que é possível gerar partituras no sistema de musicografia braile, tanto para exercícios técnicos, como para músicas simples. Embora alguns elementos da musicografia não possam ser representados em um arquivo MIDI, é possível gerar uma partitura com uma taxa de acerto acima de 90%. Desta forma, a ferramenta desenvolvido apresenta-se como uma alternativa gratuita em relação aos softwares comerciais para geração de partituras braile.

Todavia, os objetivos deste trabalho foram alcançados, ainda que o processamento de alguns elementos da notação musical não tenha sido implementado.

4.1 LIMITAÇÕES

Durante o desenvolvimento do trabalho alguns problemas puderam ser verificados e são agora relacionados. Tratando-se de uma primeira versão, este trabalho foi desenvolvido de maneira a atender os objetivos propostos. Portanto, nem todas as situações encontradas em uma partitura de tinta foram tratadas.

A identificação de quiálteras não foi implementada. Quiáltera é o nome dado a qualquer alteração na subdivisão da duração das notas. Por exemplo, se em um tempo cabem duas colcheias e escrevemos três, cada uma valendo um terço do tempo, tem-se uma tercina.

A ferramenta identifica notas com ponto de aumento e ligadura de duração, além das figuras rítmicas simples. A identificação de quiálteras é sensivelmente mais complicada, e por esse motivo, somando-se o prazo curto para a entrega, optou-se pela não implementação desta funcionalidade.

Outro ponto que não foi implementado, é a identificação dos acordes, ou seja, mais de duas notas sendo tocadas ao mesmo tempo, respeitando regras de intervalo musical. Foi verificado que essa identificação não é complexa, porém, envolve alterações no processamento do arquivo MIDI e consequentemente, na tradução para braile. A duração das notas e todo o resto do cálculo feito permanecem inalterados. Novamente, a questão do prazo foi determinante para a não implementação desta funcionalidade. Ainda assim, esta limitação afeta somente instrumentos que permitem tocar mais de uma nota simultaneamente.

Nem todas as situações possíveis em uma partitura foram testadas, portanto podem ocorrer outras situações que não foram abordadas na especificação deste trabalho e que não foram aqui citadas nas limitações.

4.2 EXTENSÕES

Algumas sugestões de extensões deste trabalho são listadas a seguir:

a) implementar a identificação de quiálteras, anacruses e acordes;
b) implementar a identificação e o processamento de pautas para piano e outros instrumentos de duas pautas;
c) implementar a execução do arquivo MIDI em conjunto com a partitura braile;
d) implementar a visualização da partitura braile em conjunto com a de tinta;
e) implementar o processamento de arquivos MusicXML.

REFERÊNCIAS BIBLIOGRÁFICAS

ALMEIDA, Marina S. R. **Manual informativo sobre inclusão**. São Paulo: [s.n.], 2004.

BOHN, Débora F. **Ensino de violino voltado para deficientes visuais integrando o método Suzuki e a musicografia braile**. 2008. 49 f. Trabalho de Conclusão de Curso (Bacharelado em Música – opção violino) - Centro de Artes, Universidade do Estado de Santa Catarina, Florianópolis. Disponível em: <http://www.pergamumweb.udesc.br/dados-bu/000000/00000000000A/00000AFB.pdf> . Acesso em: 14 nov. 2012.

BONILHA, Fabiana F. G. **Do toque ao som**: o ensino da musicografia braile como um caminho para a educação musical inclusiva. 2010. 280 f. Tese (Doutorado em Música) – Departamento de Música, Universidade do Estadual de Campinas, Campinas. Disponível em: <http://www.bibliotecadigital.unicamp.br/document/?code=000777480>. Acesso em: 11 abr. 2012.

_____. Ensino de musicografia braile. In: CONGRESSO DA ANPPOM, 17., 2007, São Paulo. **Anais...** São Paulo: UNESP, 2007. p. 01-06. Disponível em: <http://www.anppom.com.br/anais/anaiscongresso_anppom_2007/educacao_musical/edmus_FBonilha_CCarrasco.pdf>. Acesso em: 09 jun. 2012.

BORGES, Antonio. **O que é o projeto Musibraille**. Rio de Janeiro, 2003. Disponível em: <http://www.musibraille.com.br/oquee.htm>. Acesso em: 19 abr. 2012.

DANCING DOTS. **Goodfeel Braille music translator**. Valley Forge, 2005. Disponível em: <http://www.dancingdots.com/main/goodfeel.htm>. Acesso em: 10 jun. 2012.

GUEST, Ian. **Harmonia:** método prático. São Paulo: Lumiar, 2002.

LOURO, Viviane. **Educação musical e deficiência**. [S.l.]: Estúdio Dois, 2006.

MAKEMUSIC. **What is MusicXML?** Eden Prairie, 2011. Disponível em: <http://www.makemusic.com/musicxml>. Acesso em: 19 abr. de 2012.

MASKE, Roberto C. **Protótipo de um sistema para auxiliar na composição musical usando regras de harmonia.** 2000. 124 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

MED, Bohumil. **Teoria da música**. 4. ed. Brasília: Musimed, 1996.

PENTRONICS AND OPTEC SYSTEMS. **Toccata Braille music translation software**. Glenmore Park, 2002. Disponível em: <http://www.pentronics.com.au/index_files/Toccata.htm>. Acesso em: 09 jun. 2012.

PRADO, Fabio. **Midi**. [S.l.], 2006. Disponível em: <http://www.maestrofabioprado.com.br/ livro/livro.pdf>. Acesso em: 19 abr. 2012.

SÁ, Elizabeth D. **A insustentável leveza do braile.** Salvador, 2001. Palestra proferida no I Simpósio Sobre o Sistema Braille, no dia 14 de setembro de 2001, em Salvador. Disponível em: <http://www.bancodeescola.com/ leveza.htm>. Acesso em: 19 abr. 2012.

SONICSPOT. [S.l.], 2007.
Disponível em: <http://www.sonicspot.com/guide/midifiles.html>. Acesso em: 18 dez. 2012.

TOMÉ, Dolores. **Introdução à musicografia braile.** São Paulo: Global, 2003.