Tradutor de Partitura de Padrões Rítmicos para Instrumentos de Percussão Utilizando MusicXML

Rafael Ecke Bisogno[1], Reiner Franthesco Perozzo[1]

[1]Curso de Bacharelado em Ciência da Computação – Universidade Franciscana - 97010-032 – Santa Maria – RS – Brasil

rafaelbisogno@gmail.com, reiner.perozzo@unifra.br

Abstract. *This paper presents the proposal of software that performs the translation of a traditional score from a rhythmic pattern to percussion instruments into a new language. From a file generated by a score editor in the MusicXML format, the necessary tags for the translation of the score are analyzed, classified and stored by the software. The final result is a file in html format with the new musical writing and the possibility of hearing the generated pattern.*

Resumo. *Este trabalho apresenta a proposta de um software que realiza a tradução de uma partitura tradicional de um padrão rítmico para instrumentos de percussão em uma nova linguagem. A partir de um arquivo gerado por um editor de partituras no formato MusicXML, as tags necessárias para a tradução da partitura são analisadas, classificadas e armazenadas pelo software. O resultado final é um arquivo no formato html com a nova escrita musical e a possibilidade de audição do padrão gerado.*

1. Introdução

Atualmente, as pessoas vivem numa época em que a música está cada vez mais presente no seu cotidiano. Ela as acompanha em diferentes momentos, como no carro, no trabalho, em festas e até mesmo nos momentos de lazer. A tecnologia e seus avanços têm facilitado o consumo da música, uma vez que a Internet, aparelhos de som, computadores e *smatrphones* estão acessíveis tanto para aqueles que apenas desfrutam uma boa música ou para os que a utilizam como instrumento de trabalho. Castro (2011) afirma que a música é um elemento essencial na vida humana, uma grande fonte de união do homem com seu interior, seus pares e com outras pessoas.

A crescente exposição das pessoas ao universo sonoro tem despertado o interesse pelo estudo dos diversos instrumentos musicais existentes. Nesse sentido, o ensino da música vem avançando e se aprimorando por meio de novas técnicas e utilização de ferramentas mais didáticas para seu aprendizado.

Conforme Gohn (2003), o uso de tecnologias digitais vem integrando um amplo leque de ferramentas de ensino para os mais variados tipos de aprendizado musical. Desde a musicalização infantil até a prática de instrumentos, da composição aos estudos de análise. Servem tanto ao aluno interessado em música como lazer, quanto ao estudante comprometido em tornar-se um profissional.

Em virtude desse contexto, busca-se desenvolver nesse trabalho um tradutor que transforme a escrita musical tradicional em uma escrita alternativa simples, direta e de

fácil entendimento para os estudantes. Esta forma de escrita foi criada e desenvolvida pelo percussionista Fernando do Ó, que utiliza o conceito de que a nota, quando tocada ou a pausa, quando a nota não é tocada, será escrita somente no tempo e contratempo da música, criando padrões rítmicos. O alvo desse estudo são os padrões rítmicos para instrumentos de percussão.

Com isso, não há a necessidade de que o usuário tenha um prévio conhecimento de teoria musical para realizar a leitura de uma partitura. As tradicionais partituras com claves, compassos, armaduras e notas, são substituídas por símbolos simplificados na tentativa de facilitar o entendimento, podendo abranger um maior número de pessoas interessadas em ler e escrever música.

1.1. Objetivo

O objetivo desse trabalho é projetar e desenvolver uma ferramenta que realize a tradução de uma partitura tradicional, gerada por um editor de partituras, para um novo sistema de notação e escrita musical direta. A ferramenta terá como entrada um arquivo no formato MusicXML, gerado pelo editor de partitura e, como saída, a nova escrita musical.

2. Referencial Teórico

A fim de conhecer melhor o assunto, foi realizado um levantamento bibliográfico que foi utilizado como base para o trabalho. Primeiramente é contemplado o MusicXML e, logo após, são abordados os princípios básicos e fundamentais da música. Por último, são apresentadas informações sobre o "Método Fernando do Ó", obtidas diretamente com o autor.

2.1. MusicXML

O MusicXML é um padrão de notação musical em *eXtensible Markup Language* (XML), criado para permitir que usuários de diferentes aplicações e ferramentas de edição de partituras possam trocar arquivos de música. Foi desenvolvido pela *Recordare LCC*, uma empresa que fornece softwares e serviços para a comunidade musical [CAMPAGNOLO 2009].

Segundo Good (2001), o MusicXML serve como um formato de intercâmbio para aplicações em notação musical. Esse formato suporta aplicações de análise, recolhimento de informação e execução. Porém não substitui os formatos especializados para aplicações individuais, como o *Musical Instrument Digital Interface* (MIDI). Baseado no formato MusicXML, é possível ter um tradutor de notação musical universal sem que haja perda de informação durante o processo de troca de arquivos entre diversos editores de partituras. A Figura 1 ilustra as diferentes maneiras que o MusicXML pode ser utilizado, como por exemplo, a escrita de uma música no editor de partituras *Sibelius* [SIBELIUS 2018].

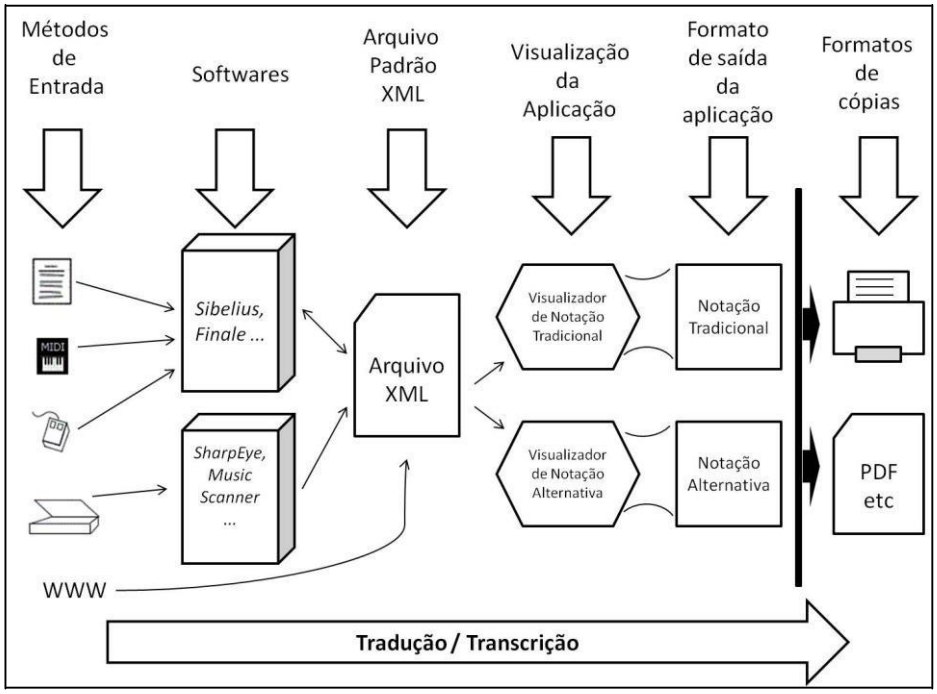

Figura 1. Exemplo utilização do MusicXML [MUSICNOTATION 2012].

Segundo Capela (2007), a Recordare desenvolveu a tecnologia MusicXML para criar um método *Internet-friendly* para publicação de pautas musicais, permitindo aos músicos que usam diferentes aplicações possam compartilhar partituras e trabalhar de forma colaborativa. Alguns programas comerciais de edição de partitura suportam o MusicXML, como o Sibellius, Finale, MusicReader, e também alguns projetos open source, como o Xemo [CUNNINGHAN 2003], LilyPond [LILYPOND 2012], Canorus(2012) e o MuseScore [MUSESCORE 2012][MILETTO 2004]. A publicação na Internet de arquivos MusicXML, pode resolver os problemas de quem necessita de um formato que garanta não só a correta representação visual da música, mas também a uma correta execução da mesma. Atualmente, o mais comum é recorrer aos serviços de download de arquivos MIDI e utilizar um conversor MIDI para pauta ou tablatura. Porém com resultados imperfeitos [FREITAS 2004].

2.2 Princípios Fundamentais da Música

A música é uma mistura harmoniosa de sons. Os sons são gerados por emissões de ondas sonoras, produzidos por diversos tipos de vibrações.

Segundo Dourado (2004), os princípios fundamentais da música como a conhecemos hoje são: ritmo, melodia e harmonia.

O ritmo é o movimento dos sons regulados pela sua maior ou menor duração. A melodia consiste na sucessão dos sons, formando um sentido musical. A harmonia é a execução de vários sons ouvidos ao mesmo tempo, observadas as leis que regem os agrupamentos dos sons simultâneos. A altura, que é a diferença entre sons graves e agudos e o timbre, que é a qualidade que permite distinguir um som do outro, são variações nas características do som [MED 2001].

A nota é um monossílabo que designa o elemento mínimo de um som. É um sinal gráfico na forma oval que representa seu valor e a altura. O valor indica a duração relativa do som e do silêncio e a altura do som da nota é determinada pela vibração, ou seja, se tiver poucas vibrações, será um som grave, e se tiver muitas vibrações, será um som agudo. Para representá-los são necessárias apenas sete notas: DÓ – RÉ – MI – FÁ – SOL – LÁ – SI, com respectivas letras correspondentes a cada nota: C – D – E – F – G – A – B [MED 2001].

O monge católico Guido d'Arezzo introduziu os monossílabos indicadores da altura do som correspondente às sete notas musicais. Os nomes das notas musicais, como são conhecidas atualmente, foram extraídos de um hino chamado Hino a São João Batista. A partir do hino, d'Arezzo nomeou as notas de acordo com o início de cada estrofe, como ilustra a Figura 2. [PORTAL 2012].

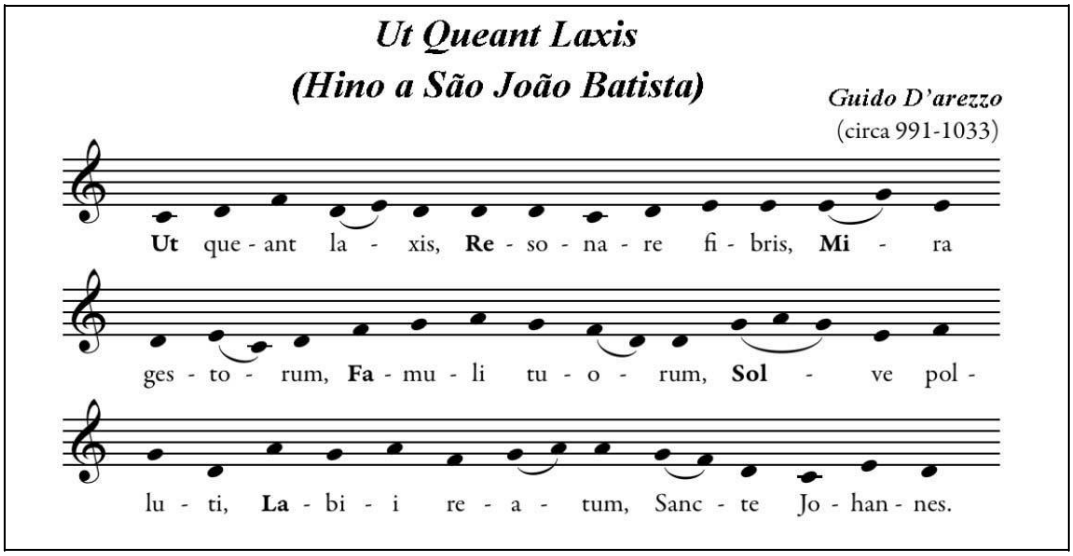

Figura 2. Hino a São João Batista.

O pentagrama, ou a pauta musical é a base sobre a qual as notas são grafadas. "Pauta é a reunião de cinco linhas horizontais, paralelas e equidistantes, formando entre si quatro espaços. São nas linhas e nos espaços da pauta que se escrevem as notas." [PRIOLLI 2007]. O nome da nota no pentagrama é determinado pela clave (em latim, "chave"), que é um sinal colocado no início do pentagrama. Cada clave dá seu nome à nota escrita em sua linha [PRIOLLI 2007].

Na notação musical, o compasso é uma forma de dividir em grupos as notas, com base em pulsos e repousos. Segundo Bennett (1984), "a música é dividida ou medida em compassos construídos com barras ou travessões".

O valor da nota indica a duração relativa do som e do silêncio. "O valor das notas, que é o tempo de duração de uma nota em relação à outra, é dado pelo formato e configuração da nota" [BENNETT 1984]. A semibreve é a figura de maior duração, portanto é tomada como unidade de divisão proporcional dos valores. Ela é a única figura que compreende todas as demais e se divide em notas de menor valor.

A primeira nota após a semibreve é a mínima, seu valor é a metade da semibreve. Logo após vem a semínima, que vale a metade da mínima. Na sequência está a colcheia, que vale a metade da semínima. A semicolcheia vale a metade da

colcheia. A fusa vale a metade da semicolcheia e a semifusa vale a metade da fusa, conforme a Figura 3. Para cada tipo de nota há um sinal correspondente chamado de pausa. Ela é o tempo de silêncio equivalente ao valor da nota [BENNETT 1984].

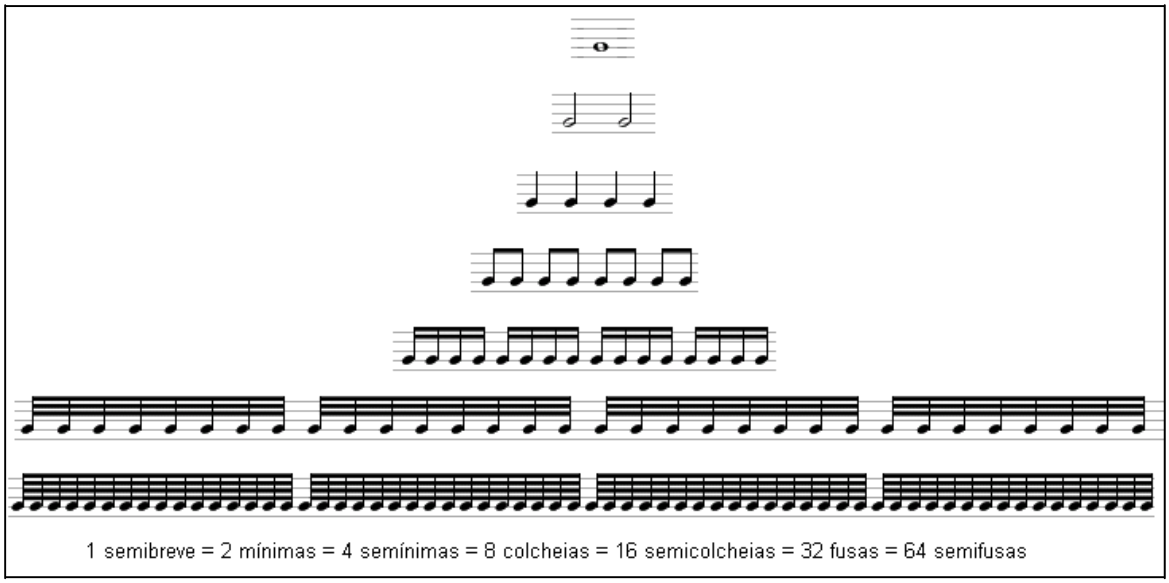

Figura 3. Tipo de notas.

2.3. Método Fernando do Ó.

Esse método de escrita musical foi desenvolvido pelo músico percussionista Fernando do Ó. Nascido em Santa Maria e hoje radicado em Porto Alegre, Do Ó já atuou em vários projetos musicais e com grandes nomes da música brasileira, como Djalma Corrêa, Ivan Lins, Adriana Calcanhoto, Orquestra de Eduardo Laje que acompanha Roberto Carlos em suas apresentações, entre outros.

Por meio de contatos com o autor do referido método, foram obtidas informações a respeito do mesmo. Nessas oportunidades, alguns pontos fundamentais com relação à formatação gráfica e funcionamento prático do método foram esclarecidos.

A nova notação é realizada escrevendo as notas em colunas dispostas uma ao lado da outra, onde cada coluna possui duas linhas. A primeira linha é o contratempo e a segunda linha é o tempo. Em cada coluna, as notas são escritas de cima para baixo, ou seja, do contratempo para o tempo. Assim, a primeira nota do compasso é escrita na segunda linha da coluna, a segunda nota é escrita na primeira linha da próxima coluna, a terceira nota na segunda linha da coluna e assim por diante como ilustra a Figura 4.

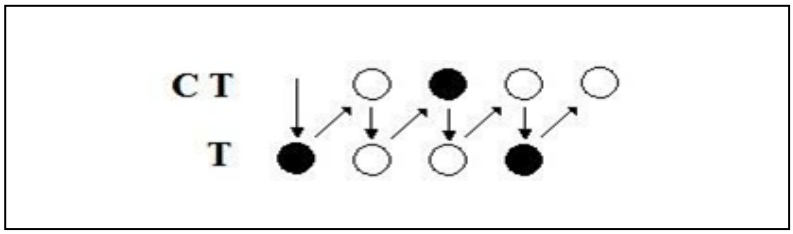

Figura 4. Funcionamento da escrita do método Fernando do Ó.

Prossegue do Ó sobre seu método, "torna-se mais prático tocar o que está marcado, ouvir o som e sentir o que está tocando, o movimento é simples e pode ser observado tanto na horizontal como na vertical". Essa forma de notação musical foi desenvolvida para ser utilizada, primeiramente, em instrumentos de percussão, porém com algumas adaptações, ela pode ser escrita para outros instrumentos.

3. Proposta

Baseado na caminhada de músicos profissionais e educadores musicais de bateria e percussão, foram observadas dificuldades que os alunos apresentam em ler e entender uma partitura convencional. A partir dessa vivência, busca-se unir a experiência acadêmica ao dia a dia desses profissionais, a fim de simplificar o ensino e aprendizagem de música.

O uso de formas não tradicionais de escrita musical é muito utilizada por músicos e educadores, visto que a escrita musical tem um papel importantíssimo no aprendizado e entendimento de música. Cada instrumento tem sua peculiaridade e a escrita musical pode ser adaptada para um melhor entendimento e aprendizagem.

A proposta deste trabalho é a criação de um *software* responsável por realizar a tradução de uma partitura tradicional para instrumentos de percussão em outra escrita e notação musical de fácil entendimento aos estudantes iniciantes. A Figura 5 apresenta uma visão geral da proposta, sendo seu fluxo descrito em seguida.

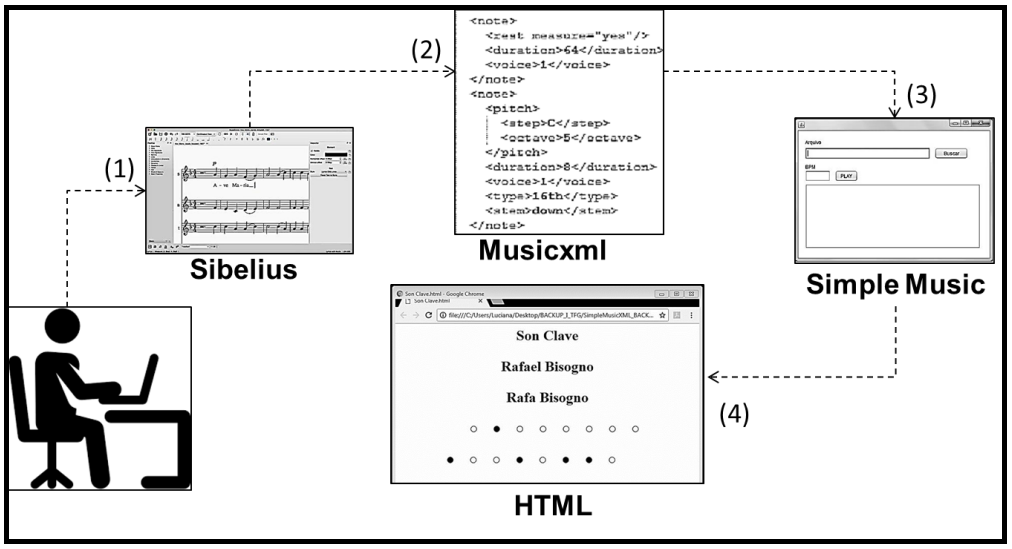

Figura 5: Visão geral da proposta

Inicialmente, o músico escreve o padrão rítmico no editor de partituras tradicional (1) e exporta o arquivo gerado no formato MusicXML (2). O aluno importa o arquivo no formato MusicXML no *software* Simple MusicXML (3) proposto neste

trabalho e que fará tradução para a nova escrita musical. O *software* gera como saída um arquivo no formato *HyperText Markup Language* (HTML) com a nova escrita musical (4).

Para facilitar a compreensão da ideia, a Figura 6 ilustra um exemplo do que realmente é obtido com o software Simple MusicXML.

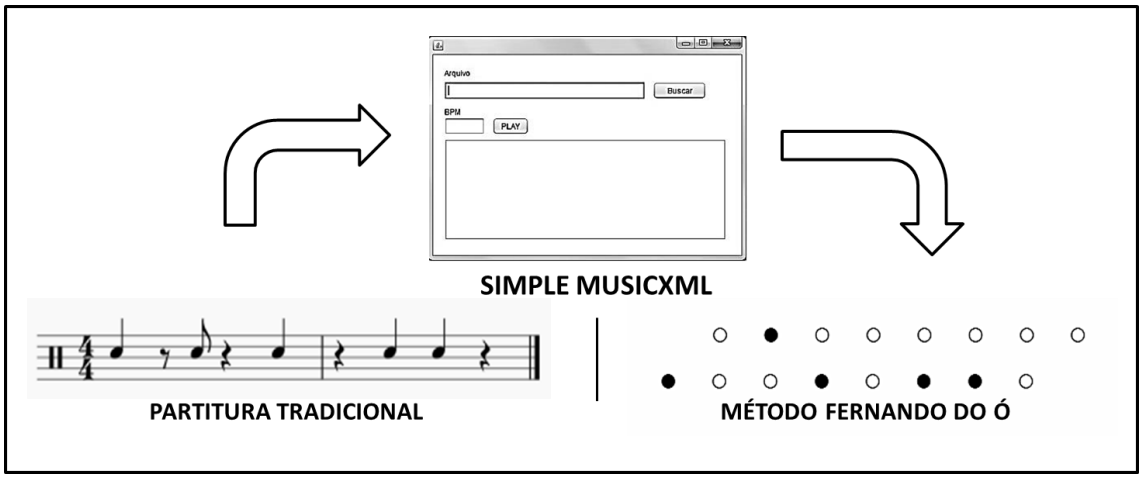

Figura 6: Exemplo de resultado obtido com o Simple MusicXML.

No que tange a metodologia de desenvolvimento de *software,* utilizou-se neste trabalho o Desenvolvimento Guiado por Funcionalidade (FDD), sendo este um modelo incremental e interativo que suporta as necessidades prioritárias do projeto [LEANDROMTR 2018]. Sua estrutura está dividida em duas fases, sendo **projeto** (análise de requisitos, documentação dos procedimentos e planejamento por funcionalidade) e **implementação** (construção por funcionalidades).

3.1 Projeto de *Software*

A partir da análise de requisitos foram identificadas e mapeadas as atividades necessárias para a construção do *software*. Na Figura 7 é apresentado o Diagrama de Classes contemplando as funcionalidades do sistema bem como a definição de atributos referentes ao desenvolvimento do mesmo. As classes que formam os componentes do *software* Simple MusicXML são: MainGUI, MusicXML, Parser, Note, Tag, Output e Player.

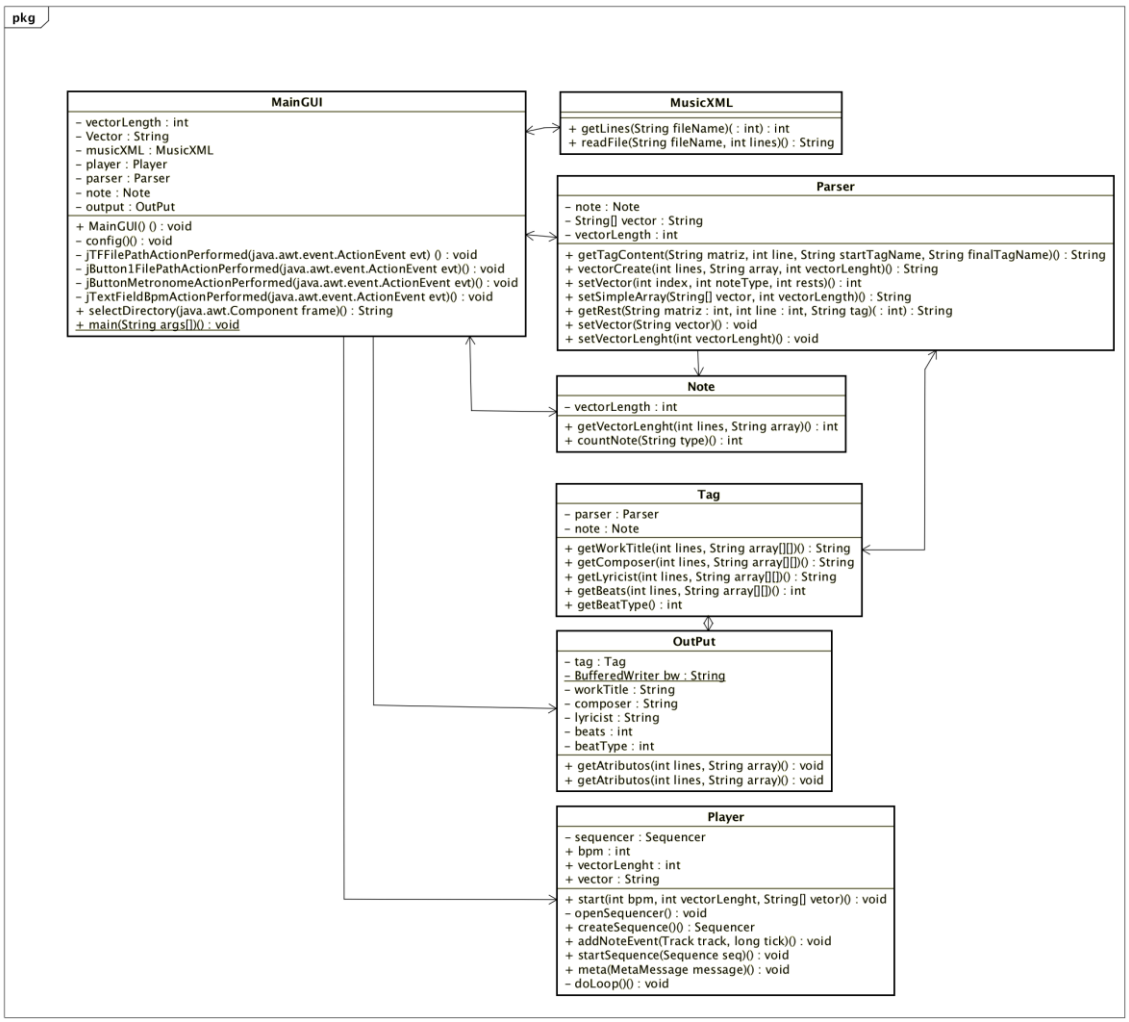

Figura 7: Diagrama de Classes do Simple MusicXML.

Considerando o Diagrama de Classes apresentado na Figura 7, há a classe MainGUI que é a responsável pela criação da interface gráfica do *software* e utilização de métodos para tradução da partitura original e apresentação da partitura simplificada, incluindo a execução sonora do padrão rítmico traduzido. Na classe MusicXML estão os métodos para a leitura e armazenamento do arquivo importado no formato MusicXML.

A classe Note possui métodos para determinar e configurar os elementos necessários, de acordo com a duração de cada nota, para a tradução da partitura. Na classe Parser ocorre a tradução propriamente dita. Os métodos buscam as *tags* do arquivo MusicXML originalmente criado pelo músico, as quais são imprescindíveis para a tradução, sendo criada uma matriz com os elementos modificados. A classe Tag é responsável por realizar as buscas das *tags* dentro da matriz criada a partir do arquivo MusicXML importado.

Realizada a tradução, a classe Output se encarrega de gerar o resultado do padrão musical criado, buscando os atributos na matriz gerada que foi criado a partir do arquivo MusicXML e criando o arquivo no formato HTML (escolhido por ser facilmente visualizado em qualquer *web browser* presente em dispositivos

computacionais). A classe Player, por fim, cria uma sequência cíclica do padrão rítmico traduzido, que pode ser ouvido pelo estudante e, também, obtido como base para comparação entre o que o aluno está tocando e o que realmente deveria tocar.

3.2 Implementação

Nesta etapa, o sistema é codificado a partir do projeto de *software*, em que se torna possível a geração da ferramenta Simple MusicXML. Dessa forma, são apresentadas a seguir, algumas das principais implementações para cada classe do projeto.

A Figura 8 exibe um trecho de um arquivo original (no formato MusicXML) que serve de exemplificação das informações que a ferramenta irá extrair para realizar a tradução. A *tag* (1) "<work-title>" é o nome da composição, (2) "<creator type="composer">" o compositor, (3) "<creator type="lyricist">", o letrista. (4) "<type>" é o tipo da nota e (5) "<\rest>" é a *tag* que indica a presença da pausa.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE score-partwise PUBLIC "-//Recordare//DTD MusicX
<score-partwise version="3.1">
  <work>
    <work-title>Clave </work-title>                    1
  </work>
  <identification>                                          2
    <creator type="composer">Rafa Bisogno</creator>
    <creator type="lyricist">Rafael Bisogno</creator>
  </identification>                                         3
  <part id="P1">
    <note default-x="76.67" default-y="-10.00">
      <type>quarter</type>
    </note>
    <note default-x="252.07" default-y="-10.00">
      <type>eighth</type>                              4
    </note>
    <note>
      <rest/>                                          5
      <type>quarter</type>
    </note>
```

Figura 8. Exemplo de código MusicXML.

Já na Figura 9, é ilustrado o método da classe MusicXML que realiza a leitura do arquivo original, no formato MusicXML, e faz a criação da matriz que servirá como base para a tradução no formato simplificado. O método *readFile()* percorre todo o arquivo MusicXML importado, linha por linha, para uma matriz base. A ideia básica é ler o arquivo original MusicXML, armazenando todo o seu conteúdo em uma matriz temporária cujo objetivo é facilitar a busca de *tags* necessárias para a tradução.

```
public String[][] readFile(String fileName, int lines) {
    String matriz[][] = new String[lines][1];
    BufferedReader br;
    String line = "";
    int i=0;
    try {
        br = new BufferedReader(new FileReader(fileName));
        while ((line = br.readLine()) != null) {
                matriz[i][0] = line;
                i++;
        }
        br.close();
    } catch (Exception e) {
        System.out.println(e);
    }
    return matriz;
}
```

Figura 9. Método readFile.

Na Classe Note, há métodos responsáveis pela tradução da partitura. O método "getVectorLenght" na Figura 10a percorre toda matriz base buscando pela *tag* "<type>". Quando a *tag* é encontrada, o método "countNote" (Figura 10b), classifica a nota de acordo com a duração da mesma, 2 para "quarter" e 1 para "eighth", e retorna o valor correspondente para a determinada nota. Será criado um vetor para o armazenamento das notas. O somatório de todos os respectivos valores das notas encontradas no arquivo MusicXML será o tamanho desse vetor, o qual varia de acordo com o numero de notas e de compassos de cada partitura tradicional.

```
public int getVectorLenght(int lines, String array[][]){
    Tag tag = new Tag();
    for (int i = 0; i < lines; i++) {
        String tagContent = tag.getTagContent(array, i, "<type>", "</type>");
        if (!tagContent.equals("")) {
            vectorLenght += countNote(tagContent);
        }
    }
    return vectorLenght;
}
```

(a)

```
public int countNote(String type){
    if (type.equals("quarter")){
      vectorLenght+=2;
      return 2;
    }
    else if(type.equals("eighth")){
        vectorLenght+=1;
        return 1;
    }
        return 0;
}
```

(b)

Figura 10. Parte do código da Classe Note.

Já na Figura 11a há o método "setVector", da classe "Parser", que recebe como parâmetro o índice do vetor criado anteriormente, o valor inteiro referente ao tipo da nota e outro valor inteiro respectivo a presença ou ausência de pausa. Esse método realiza o preenchimento do vetor de acordo com os dados recebidos na análise da matriz base. Depois de realizada essa etapa, ainda na classe "Parser", o método "setSimpleArray" configura a matriz de saída denominada "outArray", Figura 11b, com os dados armazenados no vetor. A matriz de saída possui duas linhas e o número de colunas é a metade mais um do tamanho do vetor.

```
public int setVector(int index, int noteType, int rests){
    if (noteType == 1 && rests == 0){
        vector[index] = "X";
        index++;
        return index;
    }
    else if (noteType == 1 && rests == 1){
        vector[index] = "*";
        index++;
        return index;
    }
    else if (noteType == 2 && rests == 1){
       vector[index] = "*";
       index++;
       vector[index] = "*";
       index++;
       return index;
    }
    else  {
       vector[index] = "X";
       index++;
       vector[index] = "*";
       index++;
       return index;
    }
}
```

(a)

```
for(int i = 0, j = 0; j < column; j++){

    if ( i == 0 && j ==0 ){
        outArray[i][j] = " ";
        i++;
    }
    if (j == (column-1) && i == 1){
        outArray[i][j] = " ";
        j++;
    }
    if (i == 0){
        outArray[i][j] = vector[index];
        index++;
        i++;
        j--;
    }
    else if (index != vectorLength){
        outArray[i][j] = vector[index];
        index++;
        i--;
    }
}
return outArray;
```

(b)

Figura 11. Tradução da partitura original para a simplificada.

Considerando a codificação apresentada na Figura 11, para que a matriz de saída fique de acordo com o "Método Fernando do Ó", é realizado o incremento e decremento dos índices da matriz, iniciando na segunda linha da primeira coluna e depois passando para a primeira linha na segunda coluna. Depois para a segunda linha da segunda coluna e assim sucessivamente até o final do vetor que terminará na primeira linha da última coluna, como apresentado anteriormente na Figura 4.

Com a matriz de saída armazenada, é gerado o arquivo de saída no formato HTML pelo método "createFile", (Figura 12), da classe "Output". As *tags* que contém os principais atributos da partitura como Título, Compositor e Letrista são obtidos por meio do método "getAtributos". O título da partitura tradicional dá o nome para o arquivo de saída. Cabe ressaltar que o formato HTML usado para a apresentação da partitura simplificada gerada, foi escolhido pela sua possibilidade de visualização em diversos dispositivos computacionais que possuam um *web browser*.

```
public void createFile(String[][] outArray) {
    try {
        bw = new BufferedWriter(new FileWriter(workTitle + ".html"));
        bw.write("<html>");
        bw.write("<h1 align='center'>");
        bw.write(workTitle);
        bw.write("<p h3 align='center'>");
        bw.write(composer);
        bw.write("<p \t h3 align='center'>");
        bw.write(lyricist);
        bw.write("<p \t h1 align='left'>");
        for (String item : outArray[0]) {
            bw.write(item);
            bw.write("     ");
        }
        bw.write("</p>");
        bw.write("<p h1 align='left'>");
        for (int coluna = 0; coluna < outArray[0].length; coluna++) {
            bw.write(outArray[1][coluna]);
            bw.write("     ");
        }
        bw.write("</p>");
        bw.write("</h1>");
        bw.write("</html>");
        bw.close();
    } catch (IOException e) {
    }
}
```

Figura 12. Implementação do método createFile.

Com a partitura traduzida, a classe “Player” é a responsável por implementar o reprodutor sonoro do padrão rítmico, criando um *loop* com todas as notas e as pausas existentes. É criado um evento para cada nota tocada, conforme o código da Figura 13.

```
Sequence seq = new Sequence(Sequence.PPQ, 1);
Track track = seq.createTrack();
int x = vectorLength-1;
int i = 0;
while("*".equals(vector[x])){
    i++;
    x--;
    if ( x == 0){
        break;
    }
}
x = i;
for ( i =0 ; i < vector.length; i++) {
    if (vector[i].equals("X")){
        addNoteEvent(track, (i+x)+1);
    }
}
return seq;
```

Figura 13. Criando o loop com o padrão ritmico.

O usuário deverá informar o BPM (Batidas Por Minuto) para a execução do padrão rítmico, isto é, o andamento em que ele será executado. Para calcular o BPM, o valor dado pelo usuário é multiplicado por dois, visto que as notas podem estar no tempo ou no contra-tempo do padrão rítmico.

Já na Figura 14, é apresentada interface gráfica do *software* Simple MusicXML. Para iniciar a tradução do arquivo MusicXML (partitura original), o usuário deverá informar a localização do arquivo no formato MusicXML. Após essa etapa, a ferramenta exibe a tradução da partitura para o padrão rítmico (1). Para o usuário ouvir o padrão rítmico gerado, basta adicionar no campo “BPM” (3) o número de batidas por minuto e acionar no botão “PLAY”.

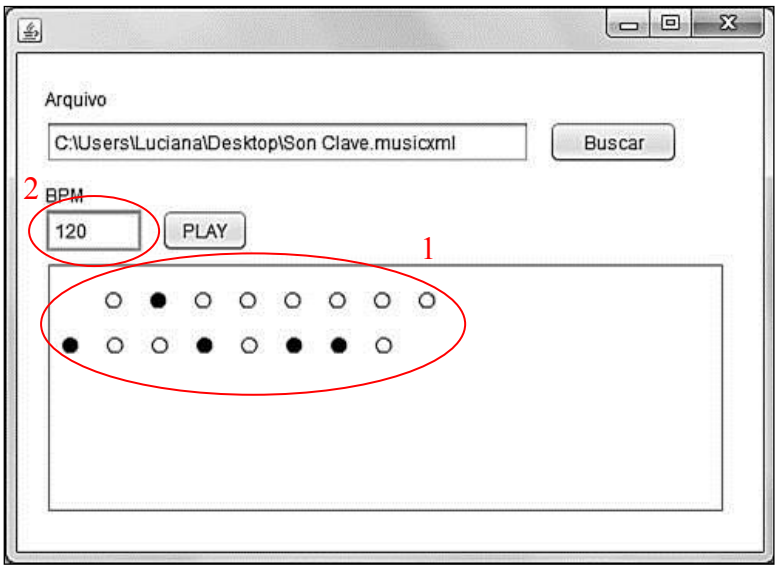

Figura 14. Layout final do Software com o arquivo traduzido

4. Resultado

Como forma de validação do trabalho e para assegurar que a ferramenta desenvolvida atenda os objetivos da proposta, optou-se por construir um cenário de validação a partir de um exemplo a tradução de um padrão rítmico muito utilizado na música cubana, a "Son Clave".

Dentro desse contexto, o Musescore 2 foi a ferramenta utilizada para gerar a partitura tradicional/original do padrão rítmico, visto que tal ferramenta é muito utilizada por músicos, arranjadores e professores. Na Figura 15 é possível visualizar o padrão rítmico criado nesse editor de partituras.

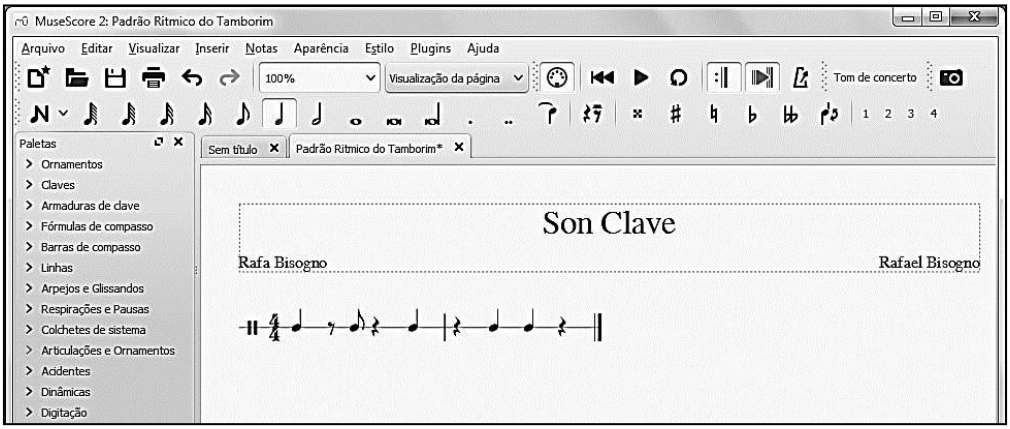

Figura 15. Padrão rítmico criado no MuseScore 2.

O próximo passo no processo da validação da proposta é exportar o padrão rítmico (do editor de partituras) para o formato MusicXML. A Figura 16 exibe parte do arquivo exportado no formato MusicXML.

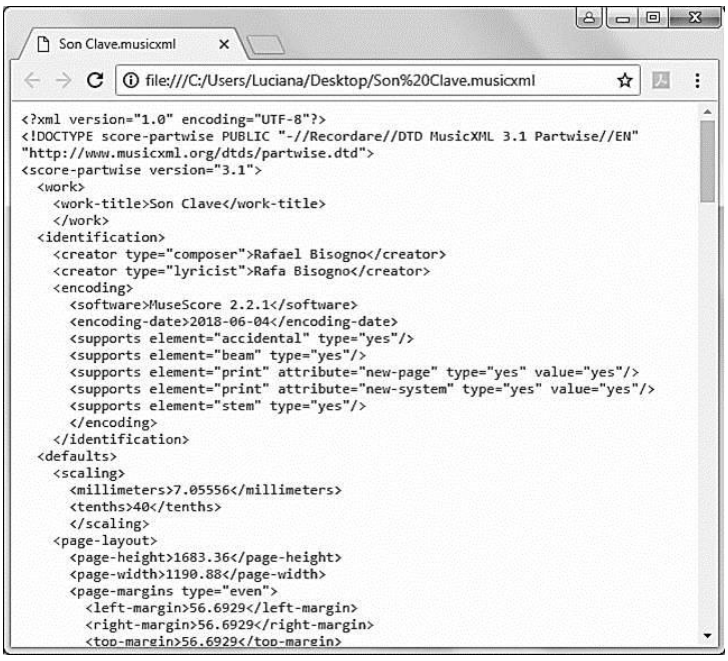

Figura 16. Arquivo no formato MusicXML.

A partir do arquivo original gerado, a ferramenta Simple MusicXML importa e realiza a tradução do padrão rítmico tradicional (que está no formato MusicXML) exibindo para o aluno o padrão rítmico traduzido para o "Método Fernando do Ó", como apresenta a Figura 17. Além disso, o usuário também pode ouvir o padrão rítmico gerado, adicionando o BPM desejado.

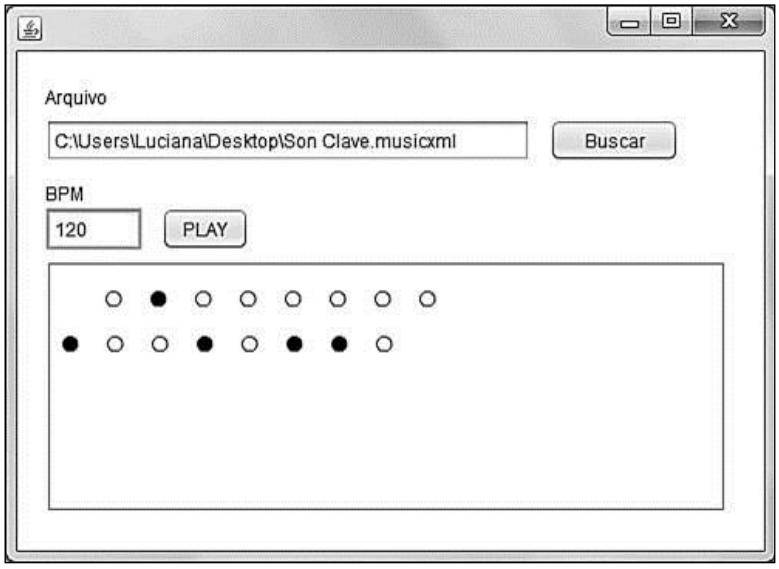

Figura 17. Padrão rítmico traduzido.

Simultaneamente a essa etapa, é gerado um arquivo no formato HTML que, além de apresentar o padrão rítmico gerado, exibe os atributos da partitura como título e compositores. Em qualquer *web browser* é possível visualizar o padrão rítmico (Figura 22).

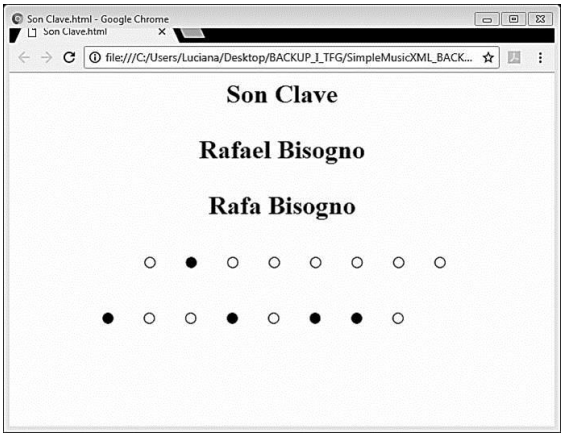

Figura 18. Arquivo no formato HTML contendo o padrão rítmico.

Tal estudo de caso serve como exemplo de validação da proposta e atende todas as expectativas levantadas na análise de requisitos, uma vez que as *tags* "<work-title>", "<creator type="composer">", "<creator type="lyricist">", "<type>" e "<rest/>", indispensáveis para a realização da tradução, estão contidos em todos os arquivos exportados pelos editores de partituras para o formato MusicXML.

5. Conclusão

Com o desenvolvimento deste trabalho, foi oportunizado o contato com o criador de uma nova metodologia de escrita e notação musical, o músico Fernando do Ó. Essa escrita visa facilitar o aprendizado de música, não havendo a necessidade de conhecer a teoria musical para desenvolver alguma habilidade. Com isso, foi proposta a criação de uma ferramenta, Simple MusicXML, para realizar a tradução de uma partitura tradicional para essa escrita simplificada.

Para entender como é feita a leitura/escrita de uma partitura tradicional, foi realizado o estudo sobre os princípios básicos e fundamentais da música. A partir desse ponto, o trabalho estabeleceu, por meio de linguagens computacionais, um elo entre a pauta musical e o esquema simplificado do "Método Fernando do Ó".

O MusicXML mostrou ser um formato capaz de realizar a integração entre diversos editores de partituras. Além disso, por ser um arquivo no padrão XML, consegue representar a notação musical para qualquer instrumento mantendo suas características naturais.

Este trabalho, por sua vez, despertou a motivação de aprofundar o estudo sobre o desenvolvimento de ferramentas e métodos que auxiliem na educação musical, a fim de atingir um número maior de pessoas envolvidas com a música, sejam elas estudantes, educadores ou profissionais.

Para trabalhos futuros sugere-se a ampliação da ferramenta proposta, com a adição de elementos que possibilitem aos usuários criarem seus próprios padrões rítmicos. Pode-se também ampliar a funcionalidade da interface gráfica, adicionando botões para a criação de padrões rítmicos e legendas para tipos diferentes de toques, utilizando o *software* de forma mais intuitiva e eficaz. Também seria interessante um aprofundamento no estudo do MusicXML, fazendo com que a ferramenta interaja com outras plataformas, viabilizando ao usuário criar suas partituras a partir de uma aplicação *web* ou de um dispositivo móvel.

Referências

BENNETT, Roy. Elementos Básicos da Música. Zahar. 1984.

MED, Bohumil. Teoria da Música. São Paulo. 2001.

CAMPAGNOLO, Fernando Quatrin. O uso de MusicXML em instrumentos de percussão: Bateria. Projeto de trabalho final de graduação. UNIFRA. 2009

CANORUS, disponível em < http://canorus.sf.net>, acesso em Junho de 2012.

CAPELA, Guilherme A. C. Sistema automático de reconhecimento de pautas musicais manuscritas. Relatório do Estágio Curricular da MIEIC 2006/2007. Porto, 2007.

CASTRO, Lincoln F.O; Bohadana, Esterlla. Educação musical e o ouvir crítico na Internet. Universidade Estácio de Sá, Programa de Pós-Graduação em Educação. 2011.

CUNNINGHAN,Stuart.MusicFileFormats&ProjectXemo.MScMultimedia Communications, 2003.

DOURADO, Henrique A. Dicionário de termos e expressões da música. São Paulo: Editora 34, 2004.

FREITAS, David. AMARAL, Jorge. Música e o XML. Processamento Estruturado de Documentos. Faculdade de Engenharia da Universidade do Porto. 2004.

GOHN, Daniel M. Auto-Aprendizagem Musical: Alternativas Tecnológicas. Annablume. 2003.

GOOD, M. MusicXML: An Internet-Friendly Format for Sheet Music. Proceedings of XML 2001 International Conference. Orlando, USA, 2001.

LEANDROMTR, disponível em http://www.leandromtr.com/gestao/metodologia-agil-fdd/ acessado em Junho de 2018.

LILYPOND, disponível em <http://lilypond.org/>, acesso em Junho de 2012.

MILETTO, Evandro M et AL. Introdução à Computação Musical – Universidade Federal do Rio Grande do Sul (UFRGS). Porto Alegre, 2004.

MUSESCORE, disponível em http://MuseScore.org/pt-br/manual-pt-br/formato-de-arquivos acessado em Maio de 2012.

MUSICNOTATION, disponível em http://musicnotation.org/software/strategy.html acessado em maio de 2012.

PORTAL, disponível em <http://www.portaledumusicalcp2.mus.br>, acesso em junho 2012.

PRIOLLI, Maria L. de M. Princípios básicos da música para a juventude. Casa Oliveira de músicas LTDA, 2007.

SIBELIUS, disponível em < http://www. http://www.avid.com/sibelius>, acesso em junho de 2018.