Universidade Federal de Pernambuco
Centro de Informática
Graduação em Ciência da Computação

Ferramenta Integrada de Aquisição e Análise do Violão Brasileiro

Valmir André de Sena
Trabalho de Graduação

Recife
2008

Universidade Federal de Pernambuco
Centro de Informática
Graduação em Ciência da Computação

Valmir André de Sena

Ferramenta Integrada de Aquisição e Análise do Violão Brasileiro

Monografia apresentada ao Centro de Informática da Universidade Federal de Pernambuco, como requisito parcial para obtenção do Grau de Bacharel em Ciência da Computação.

Orientador: *Geber Lisboa Ramalho*

Recife
2008

Assinaturas

Geber Lisboa Ramalho (Orientador)

Valmir André de Sena (Aluno)

Dedico este trabalho à minha
esposa que tanto me apoiou
durante todos os momentos difíceis
vividos durante a graduação

Agradecimentos

Agradeço primeiramente a Deus que me conduziu durante essa difícil caminhada, a minha esposa que suportou muita noite sozinha em casa enquanto eu fazia projetos no CIN. Ao meu orientador Geber que me guiou nos caminhos deste projeto. Aos meus colegas que viraram noites comigo fazendo projetos CIN. E a todos os professores do CIN que ajudaram na minha formação.

Resumo

Várias ferramentas têm sido desenvolvidas no Centro de Informática da Universidade Federal de Pernambuco na área de computação musical para o estudo do fenômeno interpretação do violão e do ritmo bossa nova, tais como ferramentas para extração de ritmo e acordes. Contudo estão ferramentas não estão integradas e necessitam de uma interface gráfica para facilitar o seu uso.

O objetivo deste trabalho é integrar estas ferramentas e criar para as mesmas uma interface gráfica amigável que facilite o seu uso, acelerando assim, as pesquisas na área de computação musical.

Palavras-chave: integração, interpretação, violão, computação musical e bossa nova.

Sumário

Lista de Ilustrações

1. Introdução

O estudo da música tem sido quase que exclusivamente dedicado à ocidental para piano. Instrumentos como o violão e o ritmo bossa nova têm sido muito pouco estudados, apesar da sua riqueza rítmica e harmônica. No estudo da interpretação musical que é um ponto estudado pela maior parte dos pesquisadores, nota-se que os músicos introduzem modificações sobre a música que está descrita na pauta, colocando nesta seu toque pessoal, o estilo. Assim, para tornar a execução musical mais interessante, os músicos variam parâmetros como aceleração, dinâmica, articulação das notas e etc.

Para se fazer um estudo da interpretação musical de forma representativa é necessário à coleta e a análise de uma grande quantidade de dados musicais, a fim de evitar questionamentos sobre os resultados. Todavia os dados musicais são bastante complexos e cheios de variáveis, então analisar-se uma grande quantidade de dados manualmente, sem auxílio de um computador, se torna inviável e pode levar muito tempo. Um caso notável foi a pesquisa sobre a obra de Charlie Parker realizada por Owens [Owens, 1947 apud [Trajano, 2007] de forma manual que levou cerca de 16 anos para ser concluída. Para a análise de uma grande quantidade de dados de forma automatizada, a área de mineração de dados tem se mostrado bastante promissora. De casos de estudos como o realizado por Owens, nota-se a importância de tais ferramentas de análise automática na pesquisa em interpretação musical.

Tendo em vista a falta de estudos sobre interpretação musical no uso do violão e do nosso ritmo bossa nova, o centro de informática da Universidade Federal de Pernambuco vem desenvolvendo diversos trabalhos na área de estudo dos ritmos bossa nova e MPB, e especialmente sobre o violão brasileiro. Dentro destas pesquisas foram criadas diversas ferramentas de captação e analise de dados, coletados a partir da execução musical de clássicos da nossa música em canções de João Gilberto, Caetano Veloso, Tom Jobim etc, por meio de um violão MIDI. Infelizmente apesar de possuírem objetivos correlatos, estas ferramentas foram confeccionadas por diferentes pesquisadores, sem integração entre si, sem interface com o usuário, e

espalhadas em diversas partes do código, o que torna a pesquisa com estas ferramentas mais trabalhosas do que deveria ser.

Dentre estas ferramentas se destacam o COCHONUT desenvolvido por Scholz em sua dissertação de mestrado [Scholz, 2007], para extrair acordes de arquivos MIDI; uma ferramenta para descoberta de padrões rítmicos desenvolvida por Trajano em sua tese de doutorado [Trajano, 2008]; e as ferramentas de microanálise desenvolvidas pelos pesquisadores de mestrado Fúlvio Figueirôa e Raphael Holanda.

A proposta deste trabalho é integrar todas estas ferramentas existentes, tornando-as de fácil utilização e extensão, fornecendo para isso uma interface gráfica com fácil acesso a todas elas. Para isto foi realizado um estudo detalhado de cada ferramenta, localizando-se as classes de maior relevância e na integração foram localizados os possíveis conflitos de classes de mesmo nome na união dos subprojetos, criadas classes de controle onde estas eram ausentes e concentrando acesso a todas elas numa classe fachada. Foi ainda adicionado o suporte a Internacionalização na interface gráfica, possibilitando o uso do programa em diversos idiomas.

1.1 Organização do trabalho

Este trabalho está organizado da seguinte forma: o capítulo 2 apresenta os conceitos e problemas relativos ao estudo da interpretação musical. No capítulo 3 são explicados os funcionamentos das principais ferramentas existentes. No capítulo 4 são descritos os passos da integração, a arquitetura e a interface com o usuário. O Capítulo 5 finaliza este trabalho, apresentando as conclusões obtidas e as perspectivas de trabalhos futuros.

2. Interpretação Musical

A interpretação musical é o passo no processo musical durante o qual as idéias musicais são transmitidas para um ouvinte [britânica, 2008], ou seja, é o momento no qual o interprete por meio de um instrumento (que pode ser a sua própria voz) traduz para o ouvinte o conceito que ele tem a respeito da música.

A música como uma arte interpretativa é um fenômeno relativamente recente na humanidade. Nas sociedades primitivas, a música desempenhava um papel mais de uma tradição oral, na qual cada intérprete tinha o senso de interpretar a tradição, e mais importante ele a renova e a transforma por meio de sua própria interpretação.

O desenvolvimento do executor como intérprete coincidiu com o desenvolvimento da notação musical. Porque os compositores por muitos séculos estiveram na posição de supervisionar a execução da sua música, desta forma certos aspectos da interpretação não eram notados [Britânica, 2008].

A interpretação é composta por dois componentes principais: um técnico e outro expressivo. O técnico se refere à mecânica utilizada para a execução da música de forma fluente e coordenada do som. O expressivo se refere às variações de forma intencional dos parâmetros musicais ligados a interpretação feitas pelo músico, para influenciar os resultados cognitivos e estéticos do ouvinte de forma única.

Se uma música é interpretada da forma exata como ela está grafada, ela soa de forma enfadonha, mecânica e artificial. A representação da música de forma simbólica é apenas ou anotada, é apenas um guia para o interprete, que a modifica de acordo com as suas intenções. Na execução os músicos desviam de forma delicada, mas significante do que está escrito na notação. Esses desvios seguem algumas regras que foram classificadas por *Friberg* e serão tratadas na seção 2.2**.** Trajano definiu a interpretação musical da seguinte forma: "*Interpretação musical é a arte de moldar uma peça musical de acordo com as intenções, regras e análises do intérprete."*

Ainda segundo Trajano, uma forte influência na interpretação musical é a forma como a mesma está representada, ou anotada. Podendo a notação musical ser classificada como completa ou parcial. O principal sistema de notação completa é o gráfico ocidental que utiliza símbolos grafados por uma pauta de cinco linhas, também chamada de pentagrama ou partitura. Exemplos de notação parcial são a cifra, a tablatura e o baixo cifrado. Esses tipos de classificação influenciam de acordo com o grau de liberdade que estas impõem aos músicos, pois não existe uma notação que consegue expressar fielmente a intenção do compositor em todos os seus detalhes. Na partitura, por exemplo, o intérprete possui a notação das durações das notas, mas não há indicações precisas sobre a dinâmica[1] de execução destas, há apenas uma indicação subjetiva como forte, fortíssimo, piano e pianíssimo. Já na notação cifrada o músico deve escolher além do andamento, a dinâmica, as notas e o ritmo, por falta destas informações neste tipo de notação. Há ainda o caso de ausência total de notação muito usada na música popular brasileira, e na música folclórica [Trajano, 2008, apud [Napolitano, 2003]], onde para manter a tradição cultural a música é passada de geração a geração.

Uma forte influência na interpretação está relacionada ao grau de conhecimento anterior que o intérprete possui com a obra que está sendo executada. Esse grau de conhecimento é classificado em modalidades da seguinte forma [Trajano, 2008, apud [Palmer, 1997]]: leitura a primeira vista; tocar de memória ou a partir de uma notação, obras já estudadas e preparadas anteriormente; improvisação e tocar de ouvido. Outro fator decisivo na interpretação é a própria experiência do intérprete, sendo apontada como sendo um dos fatores principais fatores que pode influenciar a interpretação.

Outros fatores que também influenciam são o papel do intérprete, a dimensão da música interpretada (melodia, harmonia e ritmo), o instrumento musical e o estilo da obra a ser executada [Trajano, 2008].

Com o que foi citado é fácil notar o grau de complexidade e o vasto campo que há para se explorar na área de interpretação musical. A figura 2.1 ilustra os passos entre a notação e a interpretação.

[1] Na música, dinâmica se refere à intensidade com que é executada uma determinada nota, por exemplo, no piano seria o quão forte a tecla é pressionada e num violão o quão forte a corda é puxada.

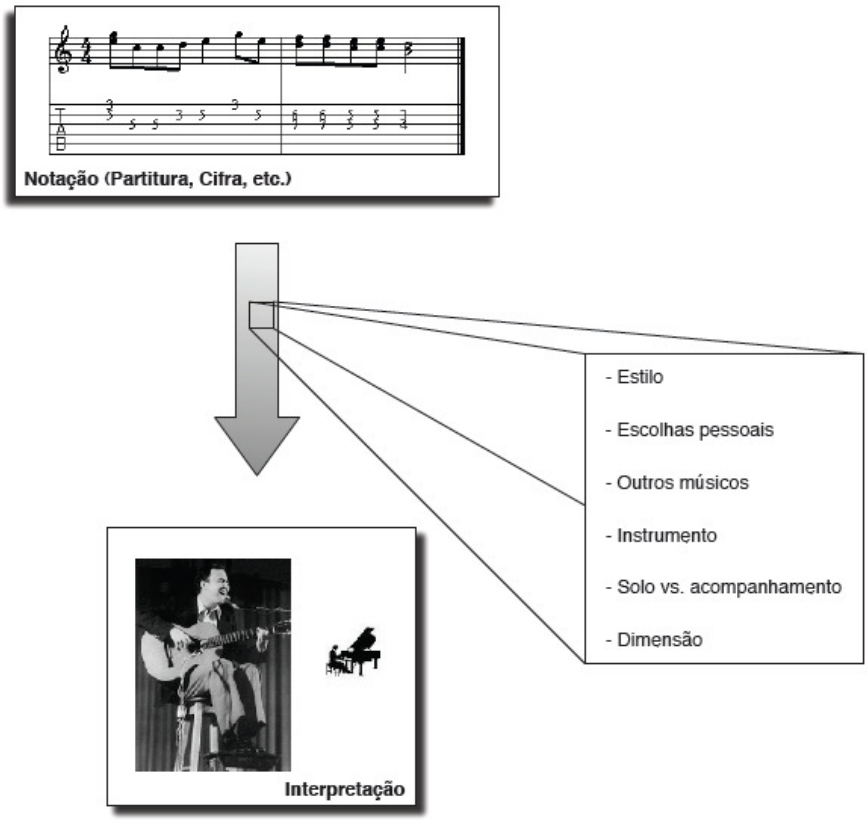

Figura 2.1 – Os passos entre a notação e a interpretação – Fonte: Trajano 2007

2.1 As pesquisas na área de interpretação musical

Em sua pesquisa sobre interpretação musical Trajano formula a seguinte pergunta: “será que a interpretação musical, sendo uma manifestação artística, não é um fenômeno intangível, que não pode, de maneira alguma, ser generalizado?” [Trajano, 2007]. Sobre o que foi previamente citado nota-se que há diversos fatores comuns entre os intérpretes. Segundo Palmer: “a interpretação é baseada tanto em elementos individuais, que diferenciam os intérpretes, quanto em aspectos normativos que são compartilhados por estes mesmos intérpretes.” [Palmer, 1997, apud [Trajano, 2007]]. Trajano coloca assim, que a interpretação é um fenômeno passivo de análise, sobre diversos aspectos e que esses são o ritmo; a dinâmica; o andamento, a articulação e a melodia.

A respeito desses aspectos de interpretação Friberg modelou uma série de regras de interpretação que são adotadas por músicos reais, para modelar

uma ferramenta (***Director Musices***) de interpretação automática a partir de uma partitura de entrada, indicando ainda que estas regras são principalmente aplicadas a música clássica ocidental, mas que podem ser aplicadas ao jazz e a música folclórica. As regras modeladas por Friberg são apresentadas na tabela 2.1 abaixo.

A. Categorias de Diferenciação	
A.1 Duration categories	A.2 Pitch Categories
Duration contrast	High sharp
Double duration	High loud
Accents	Melodic charge
	Melodic intonation
B. Regras de agrupamento	
B.1 Microlevel	B.2 Macrolevel
Punctuation	Phrase arch
Leap articulation	Phrase final note
Leap tone duration	Harmonic charge
Faster uphill	Chromatic charge
Amplitude smoothing	Final ritard
Inégales	
Repetition articulation	
C. Regras em grupo	
Melodic synchronization	
Bar synchronization	
Mixed intonation	
Harmonic intonation	

Tabela 2.1 – Regras de interpretação de Friberg

As regras são agrupadas de acordo com o papel que desempenham na comunicação da interpretação musical, sendo os grupos de regras de Diferenciação e de agrupamento as duas principais categorias [Friberg, 1995].

Conhecer o grau de aplicação de tais regras e se existem diferentes regras para diferentes estilos musicais é muito importante para se conhecer melhor o fenômeno da interpretação. (microtiming, dinamics)

Faremos uma breve descrição de algumas destas regras de Friberg na próxima seção.

2.2 Regras de Interpretação de Friberg

2.2.1 Categorias de Diferenciação - Duração

DURATION CONTRAST - Os músicos às vezes aumentam o contraste entre os valores de notas muito curtas e muito longas, tocando as curtas ainda mais curtas e as longas ainda mais longas. Uma evidência para esse princípio foi encontrada por Taguti et al. [Taguti et al, 1994, apud [Friberg, 1995]], onde foi medida a execução do terceiro movimento da sonata K. 545 para piano de Mozart e foram encontradas seções com notas com o tempo de semicolcheia, sendo executadas em menor tempo do que as que possuíam tempo de colcheia.

DOUBLE DURATION - Para duas notas tendo a taxa de duração de 2:1 a curta terá sua duração alongada e a mais longa encurtada. Foram encontrados exemplos em muitas músicas diferentes por [Gabrielsson, 1987 apud [Friberg, 1995]].

ACENTOS – *Acentos são dados a notas nos seguintes contextos*: (a) uma nota curta entre duas notas longas, (b) a primeira de várias notas curtas, e (c) a primeira nota longa depois de uma nota acentuada.

2.2.2 Categorias de diferenciação de Pitch

HIGH SHARP – Um desvio de pitch de igual temperamento é aumentado na proporção da altura do pitch.

HIGH LOUD – Esta regra aumenta a intensidade da nota na proporção da altura desta. Um dos propósitos desta regra é modelar as propriedades físicas de alguns instrumentos**.**

○

2.2.3 Microlevel Grouping

PONTUAÇÃO - A menor unidade na estrutura hierárquica musical é formada por grupos de tons de até cerca de 7 notas, chamados gestos melódicos. Ao reproduzi-las é importante marcar os limites entre estes gestos. Estes limites são marcados com micropausas.

2.3 O uso do computador na análise da interpretação musical

Como pode ser visto a interpretação musical é composta de muita complexidade, e como citado por Trajano, para que uma pesquisa em interpretação musical seja significativa é preciso uma quantidade razoável de dados, para que os modelos de interpretação desenvolvidos sejam representativos.

Todavia a análise de uma grande quantidade de dados coletados sem o auxílio de um sistema computacional é muito lento e acaba sendo desmotivante. Um caso notório da dificuldade atingida por tipo de pesquisa sem o auxílio de computador, foi a realizado sobre a obra de Clarlie Parker por Owens [Owens, 1974, apud [Trajano, 2007]], que somente finalizou a sua pesquisa de doutorado após 16 anos de pesquisa. Com a mineração de dados, a pesquisa em dados armazenados eletronicamente é automatizada ou pelo menos facilitada por meio de um computador, algo que como visto é essencial à pesquisa em computação musical.

2.4 - Modelos para extração de dados e padrões musicais

2.4.1 Extração de Ritmo

Em relação ao ritmo há três diferentes níveis de abstração da estrutura musical a partir dois quais um modelo de interpretação pode ser construído: a nota, a estrutura métrica e o grupo rítmico. [Trajano, 2007].

Ao nível da nota está relacionada à duração de cada nota, sendo a unidade mais simples da análise. A estrutura métrica está relacionada à sensação de pulsação que sentimos ao ouvir uma música. O grupo rítmico é organização natural do sinal sonoro em unidades. O nível da estrutura métrica está acima do nível da nota e o nível do grupo rítmico está acima do nível da estrutura métrica, formando assim um nível hierárquico.

Os modelos atuais em sua maioria só analisam a interpretação ao nível da nota, desprezando uma visão das notas como um conjunto. Mas na bossa nova sabe-se que o acompanhamento se dá por meio da concatenação de diversos grupos rítmicos [Garcia,1999, Sandroni, 1988 apud [Trajano,2007]]. Desta forma a análise a nível estrutural da nota não permite a identificação destes grupos. O estudo desses grupos rítmicos é de importância vital no estudo rítmico da bossa nova, pois esses grupos possuem um certo grau de recorrência no estilo.

Assim Trajano define que a modelagem do acompanhamento rítmico da bossa nova deve ser feita em dois estágios: A determinação dos padrões rítmicos utilizados pelos intérpretes; e a construção de um modelo que descreve os padrões encontrados. A seguir são discutidos os problemas inerentes as estes estágios.

2.4.1.1 Estágio de extração dos padrões musicais

Na computação a área responsável pela extração de padrões em dados armazenados eletrônica mente é a de mineração de dados, uma subárea de aprendizagem de máquina. Mineração de dados é definida como o processo de descobrir padrões em dados. [Witten, Frank 2004]. Diz-se que um padrão é estruturado se ele ajuda a explicar algo sobre os dados. Os dados podem ser estruturados classicamente como árvores de decisão, regras, agrupamentos etc. Os dados musicais são em geral representados como seqüências ordenadas que são posteriormente decompostas e analisadas de maneira tal que determinadas recorrências (os padrões) possam ser identificados [Trajano, 2007].

Como a música é composta por unidades seqüenciais (notas), um padrão musical pode assim ser considerado um subconjunto da completa seqüência musical que se repete de forma recorrente. Algo que torna a extração destes padrões bastante complexa é a que os elementos musicais são compostos por múltiplos atributos, por exemplo, uma nota possui os atributos de altura, duração, timbre, intensidade etc. Além desta complexidade existente num elemento tão simples como uma única nota, podemos ter várias notas tocadas ao mesmo tempo, cada uma com seu conjunto de atributos, para tornar as coisas ainda mais difíceis numa única música podemos ter presentes ainda vários instrumentos simultâneos.

Um problema central na identificação dos padrões é a especificação da medida de similaridade entre os padrões, já que essa medida determinará se uma subseqüência será similar à outra, ou seja, se é ou não uma recorrência de outra [Trajano, 2007]. Há outros subproblemas delimitados por Trajano, a forma como a música pode ser representada, que normalmente é feita por símbolos; e a técnica de extração dos padrões.

No que concerne à representação da música por símbolos, das várias formas existentes elas pode ser resumidas em duas representações básicas, a saber: cadeias de eventos e cadeias de intervalos [Meredith et al, 2002, apud [Trajano, 2007]]. Nas cadeias de eventos cada símbolo representa um evento

musical, e cada símbolo é colocado na cadeia de forma sucessiva, respeitando o momento que aparece na música. A cadeia de eventos tem sido empregada com sucesso em recuperação de informação musical, mas apresenta limitações para representar níveis estruturais acima da nota. Nas cadeias de intervalos os símbolos não representam os eventos, mas sim a transformação necessário para se transformar um evento em outro. A representação por cadeias de eventos facilita que ocorrências de padrões invariantes quanto a transposição sejam encontrados, o que ocorre, por exemplo, quando um mesmo trecho de uma música é tocada mais de uma vez mas em escalas musicais diferentes. Um exemplo das duas formas de representação, para o início da música desafinado, pode ser visto na figura 2.2.

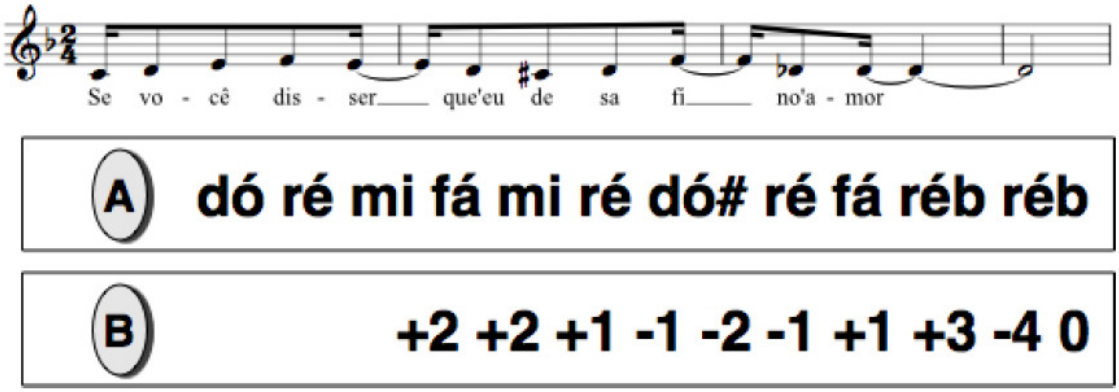

Figura 2.2 - Introdução da música desafinado. Fonte [Trajano, 2007].

Em relação à técnica de extração de padrões, certamente a mais utilizada é o casamento de padrão (matching). Essa técnica é muito utilizada para encontrar padrões em texto e em seqüências de DNA. Como algoritmos de casamento de padrões podemos citar: o Naive-string-matching, o Rabin-Karp e o KMP. Esses algoritmos se adequam muito bem a busca de padrões em texto e em seqüências de DNA, mas em música o casamento de padrões utilizando esses algoritmos que buscam seqüências exatas, principalmente no que concerne a interpretação musical não é adequado. Isto se deve ao fato que seria humanamente impossível para qualquer intérprete executar uma música exatamente da mesma maneira. Certamente os músicos variam a duração e as intensidades das notas. Uma interpretação exata só aconteceria se fosse realizada por meio de uma máquina como um computador.

Trajano aponta ainda que um dos motivos para não haver a necessidade de casamentos exatos na análise da interpretação musical, é que a maioria das repetições nas músicas do ponto de vista perceptivo não são significativas, mesmo as exatas. Assim é uma tarefa difícil desenvolver um algoritmo que encontre as repetições que são significativas, pois isto envolve caracterizar formalmente quais são interessantes do ponto de vista do ouvinte.

Se muitas das repetições significativas não são exatas, então o que pode ser feito? Usa-se ao invés do casamento exato, um casamento aproximado, de forma que certas diferenças entre os padrões possam ser ignoradas. Para essas diferenças podemos citar buracos (pausas) inseridos entre os padrões, pequenas variações no tempo e na intensidade das notas. Pesquisas mostram que muitas dessas pequenas variações são usadas propositalmente pelo intérprete para destacar as estruturas musicais [Trajano, 2007].

Com a consideração dessas diferenças surge um novo problema, até que ponto essas diferenças são aceitáveis? Pois a certo ponto as diferenças serão tantas que não haverá uma similaridade aceitável. Então devemos primeiramente estabelecer uma medida do grau de similaridade entre essas repetições e então determinar qual o grau aceitável para se considerar que duas repetições são similares. Estudos demonstram que devem ser feitas restrições na extração destes padrões de repetição, senão poderão ser pegos padrões muito longos, que tendem a duplicar informações; ou padrões muito curtos que terão muito pouca informação significativa sobre a semântica musical [Liu et al, 2005].

Uma medida de similaridade comumente usada para estabelecer a similaridade entre duas strings é à distância de levenshtein ou distância de edição, o que corresponde ao número mínimo de passos necessários para se transformar uma string em outra. Os passos correspondem às operações de inserção, substituição e deleção. Quanto maior o número passos necessários para se transformar uma cadeia na outra, maior será a distância de edição entre essas elas, e assim menor a sua similaridade. Em um estudo sobre a aplicação da distância de edição no domínio musical, propões-se ainda, que

outras duas operações poderiam ser realizadas, fragmentação e consolidação [Mougen e Sankoff, 1990 apud [Trajano, 2007]]. Na seção seguinte veremos alguns algoritmos de casamentos de padrões utilizados em computação musical.

2.4.1.2 FIExPat - Flexible Extraction of Sequential Patterns

FIExPat é um algoritmo de extração capaz de realizar casamento aproximado de padrões seqüenciais[Rolland, 1999, Rolland, 2001 apud [Trajano, 2007]]. Na literatura encontra-se muito pouco a respeito deste algoritmo, assim descrever o que foi encontrado no trabalho de Trajano. Tem origem em algoritmos da bioinformática usados para detecção de padrões em cadeias de DNA, e faz usa das distância de levenshtein, com o acréscimo das operações de fragmentação e consolidação propostas Mougen e Sankoff.

A entrada do algoritmo é formada por uma seqüência de dados simbólicos representando um sinal monofônico e sua saída é um conjunto de padrões. Cada padrão encontrado é chamado de classe, e for formado por duas partes: protótipo e conjunto de recorrências do protótipo. O protótipo é seqüência que melhor representa as similaridades no padrão encontrado numa mesma classe.

O FIExpat é dividido em duas etapas: comparação e classificação. Na etapa de comparação a entrada é dividida em subseqüências, e cada uma destas subseqüências tem a distância de edição calculada para cada uma das outras. Desta etapa é gerado um grafo chamado grafo de eqüipolência. Eqüipolência é definida da seguinte maneira: “Duas seqüências s1 e s2 são ditas eqüipolentes se e somente se, sua similaridade é maior ou igual a um dado limiar” [Trajano, 2007]. As etapas do FIExpat estão ilustradas na figura 2.3.

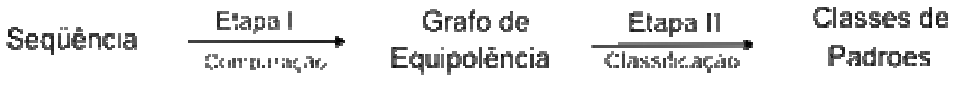

Figura 2.3 – Etapas de processamento do algoritmo FIExPat

O grafo de eqüipolência é um grafo no formato estrela com o protótipo sendo o vértice no centro e o conjunto de recorrências semelhantes sendo vértices em volta do protótipo. As arestas são rotuladas com as similaridades entre o protótipo e seu conjunto de recorrências. A figura 2.4 mostra um exemplo de um grafo de eqüipolência.

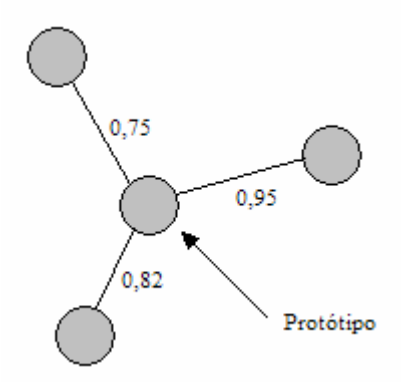

Figura 2.4 – Grafo de Eqüipolência

A seguir vemos uma descrição simplificada dos algoritmos das duas etapas do FIExPat, nos quadros 2.1 e 2.2.

Algoritmo 1 – Etapa de comparação do FIExPat

Para todos os pares de subseqüências de s, s[i]]t] e s[j][t´]
Por ordem crescente de t e t´ faça.
cDel ← S(s[i][t-1], s[j][t´]) + DEL(s[i][t]) { cálculo do custo de deleção }
cIns ← S(s[i][t], s[j][t´-1]) + INS(s[j][t´]) { cálculo do custo de inserção }
cSub ← S(s[i][t-1], s[j][t´-1]) + SUB(s[i][t], s[j][t´]) { cálculo do custo de deleção }
S(s[i][t], s[j][t´]) ← min(cDel, cIns, cSub))
Se S(s[i][t]; s[j][t0]) ≥ l { l é e o limiar de similaridade} **Então**
    Adicione uma aresta no grafo de eqüipolência entre entre estas duas subseqüências, com o valor de S calculado
fim Se

Quadro 2.1 - Etapa de comparação do FIExPat.

Algoritmo 2 – Etapa de classificação do FIExPat

Criar uma lista de padrões
Para cada vétice de padrões lp, inicialmente vazia
Compute a simVizTotal(v)
Se simVizTotal(v) > q {q é o quorum mínimo } **Então**
    Adicione a lp o maior subgrafo estrela cujo vértice central é v

Quadro 2.2 - Etapa de classificação do FIExPat

2.4.1.3 SimilaritySegmenter

SimilaritySegmenter é um algoritmo genético capaz de encontrar padrões em música representada na forma de um grafo [Madsen e Widmer, 2005, Madsen e Widmer 2006a, apud [Trajano, 2007]. O algoritmo segue a mesma idéia geral de um algoritmo genético, onde um objeto musical pode por meio de uma seqüências de operações (mutações, cruzamento e seleção) transformado em outro. Para se determinar a similaridade entre dois objetos musicais, se há um padrão, são necessárias duas etapas: Determinar quais são as operações de transformação e qual deve ser a seqüência de aplicação destas operações.

Neste algoritmo um grafo dirigido é utilizado para representar a música. Cada vértice representa um evento que pode ser uma nota ou uma pausa. As arestas representam o tipo de relação entre os vértices e podem ser de dois tipos, “segue” e “simultâneo”, respectivamente para indicar que um evento acontece após o outro, ou que os eventos acontecem simultaneamente. A figura 2.5 ilustra um exemplo deste grafo. Na figura os eventos *a* e *c* ocorrem simultaneamente e são seguidos pelos eventos *b* e *d* que também ocorrem simultaneamente.

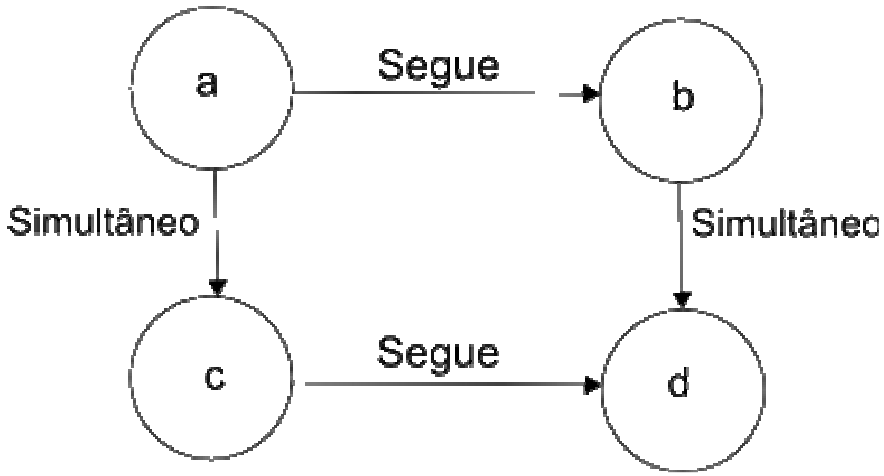

Figura 2.5 - Exemplo de parte de um grafo utilizado pelo algoritmo similarity check.

O funcionamento do algoritmo se dá seguinte forma: Uma população de ocorrências similares (OS) é inicializada aleatoriamente. Com uma OS representando uma estimativa de que dois grafos são similares. Por meio das operações usuais dos algoritmos genéticos e modificando o tamanho e a posição de cada OS, o evolui sua população, identificando grafos cada vez mais similares. Depois de atingido certo número de gerações, o algoritmo termina e o OS mais apto, determinado por uma função é comparado a um limiar para testa se nos subgrafos há similaridade suficiente para que se considerem padrões, caso isso ocorra, uma busca é realizada para outras possíveis ocorrências do padrão previamente encontrado, antes que uma nova população seja gerada e o processo reinicie.

3. Descrição dos Subprojetos

Neste capítulo detalharemos o funcionamento dos principais subprojetos.

3.1 A Correção de Ruídos em Seqüências MIDI Capturadas

A correção de ruídos das seqüências capturadas pelo violão se baseia em dois tipos de regras, aplicadas seqüencialmente: uma base de regras simples e uma base de regras contextuais. Tais regras utilizam o motor de inferência JEOPS [JEOPS apud [Scholz, 2007]].

A base de regras simples marca notas não plausíveis ou possíveis ruídos, ou removem notas claramente ruidosas. Está base contém sete regras, que são apresentadas na tabela 3.1. As regras são disparadas pelo motor de inferência de forma prioritária, isto é, as regras da parte de cima da tabela têm prioridade maior que as situadas na parte de baixo [Scholz, 2007].

Nome da Regra	Função da Regra
DeleteShortNoteEvents	**Remove notas com duração inferior a 100 *Ticks, ou 125 milésimos de segundo***
DeleteLowVelocityEvents	**Remove notas com *velocity* inferior a 15**
MarkOutOfRangeEvents	**Marca como não plausíveis eventos fora dos limites da corda do violão**
MarkPotentialNoisyEvent_SixStrings	**Marca como possíveis ruídos eventos muito distantes fisicamente, no braço do violão, de outros eventos ocorrendo simultaneamente em outras cordas do Violão**
MarkPotentialNoisyEvent_FiveStrings	
MarkPotentialNoisyEvent_FourStrings	
MarkSimultaneousEvents	**Marca como possíveis ruídos notas simultâneas numa mesma corda do violão**

Tabela 3.1 – Regras simples para minimização de ruídos.

A base de regras contextuais é aplicada após um particionamento da seqüência, processo no qual os blocos onde potencialmente há um mudança de harmonia são identificados e que será visto na seção 3.2.1. Esta base é composta de cinco regras, que são exibidas na tabela 3.2.

Nome da Regra	Função da Regra
ComplyWithPreviousChordBySimilarity	**Tenta corrigir a freqüência de notas ruidosas ou não plausíveis considerando a similaridade com o acorde vizinho**
ComplyWithNextChordBySimilarity	
ComplyWithPreviousChord	**Tenta corrigir a freqüência de notas ruidosas ou não plausíveis considerando apenas os acordes anteriores ou posteriores**
ComplyWithNextChord	
DeleteRemainingNonPlausibleEvents	**Após todas as tentativas de correção, remove as notas marcadas como não plausíveis ainda existentes**

Tabela 3.2 – Regras contextuais para minimização de ruídos.

3.2 O Reconhecimento de Acordes a Partir de Seqüências MIDI

O método usado para extração de acordes em seqüências utilizada neste projeto foi fruto da dissertação de mestrado de Scholz [Scholz 2008]. E é realizada em três etapas, a saber:

- A segmentação da seqüência por meio da identificação de pontos de particionamento que marcam as mudanças na harmonia da música;
- A escolha de um conjunto de acordes mais prováveis para cada segmento, de acordo com as informações locais;
- A escolha do acorde mais adequado a cada segmento, considerando as informações contextuais.

As seqüências midi utilizadas como dados de entrada devem possuir seis trilhas, uma para cada corda do violão, organizadas da mais aguda a mais grave da seguinte forma: a mais fina de som mais agudo (entre todas) E no canal 11, a segunda B no canal 12, e assim por diante até a corda E mais grossa no canal 16. A figura 3.1 ilustra essa configuração.

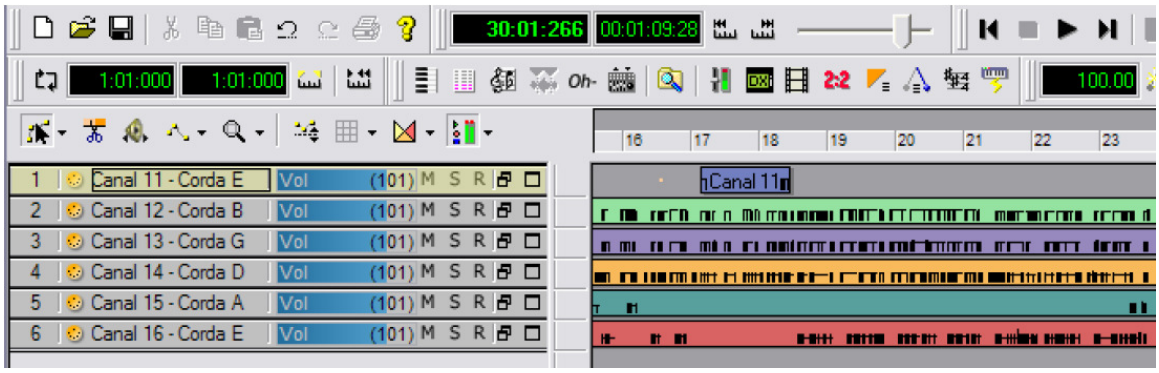

Figura 3.1 – Seqüência midi com seis trilhas, uma para cada corda do violão.

O processo utiliza várias técnicas de inteligência artificial tais como, teoria da decisão, otimização, casamento de padrões e reconhecimento baseado em regras. Além das três etapas mencionadas anteriormente o processo envolve também a etapa de limpeza dos ruídos nas seqüências MIDI gerados durante o processo de captação do violão, e que foi discutido na seção anterior. A figura 3.2 ilustra o etapas do processo de extração de acordes.

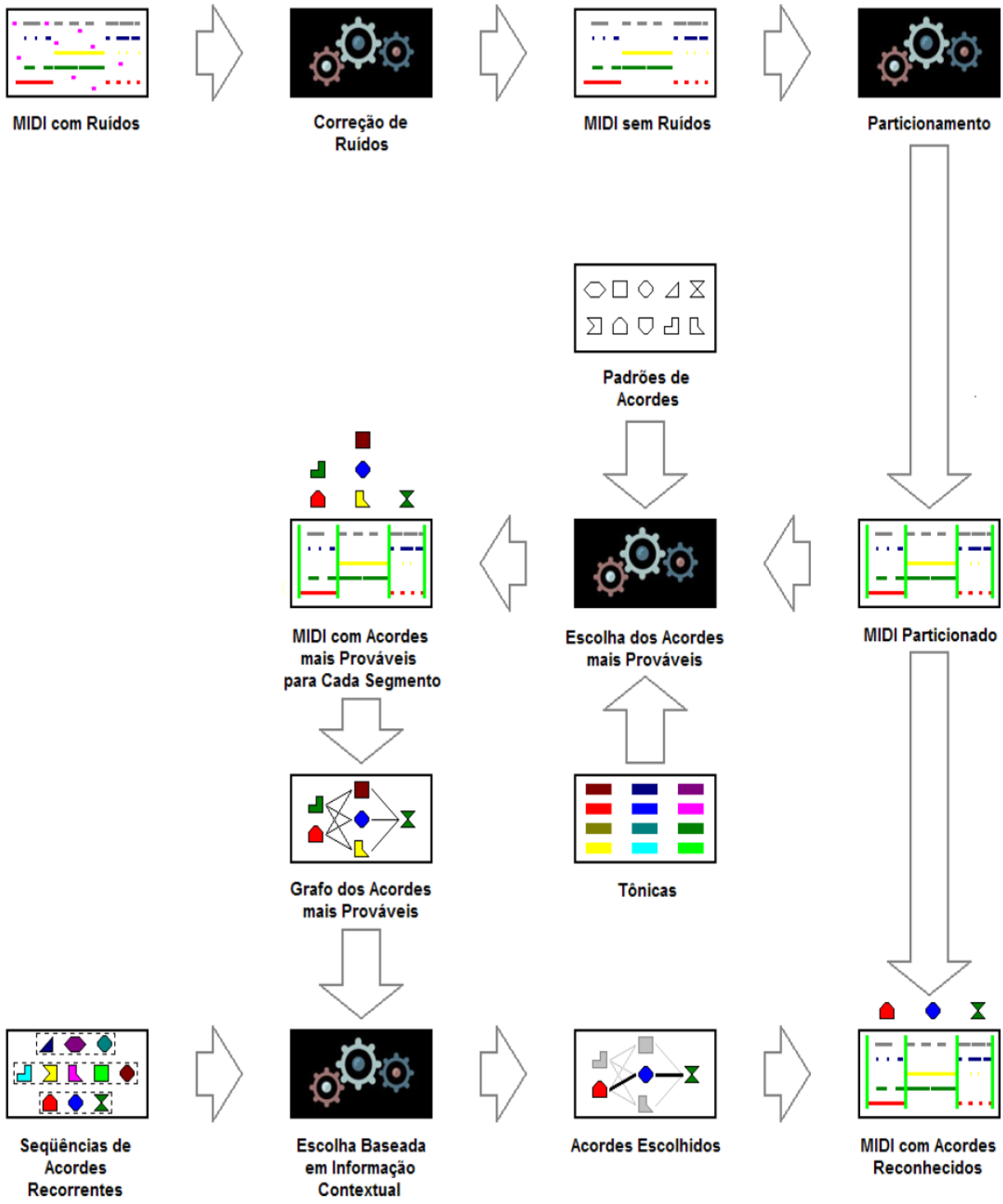

Figura 3.2 – Etapas no processo de extração de acordes do COCHONUT

Nas seções seguintes veremos de forma breve como funciona cada uma dessas etapas.

3.2.1 Segmentação das seqüências midi

A função desta etapa é definir pontos de partição onde possivelmente houve uma mudança de acorde. Entre cada par de pontos de partição é definido um segmento. É representado por um vetor de inteiros de doze elementos, chamado vetor de estados, onde cada inteiro representa o número de vezes que uma nota da escala musical temperada ocidental, apareceu no segmento. Um exemplo deste vetor pode ser visto na figura 4.3.

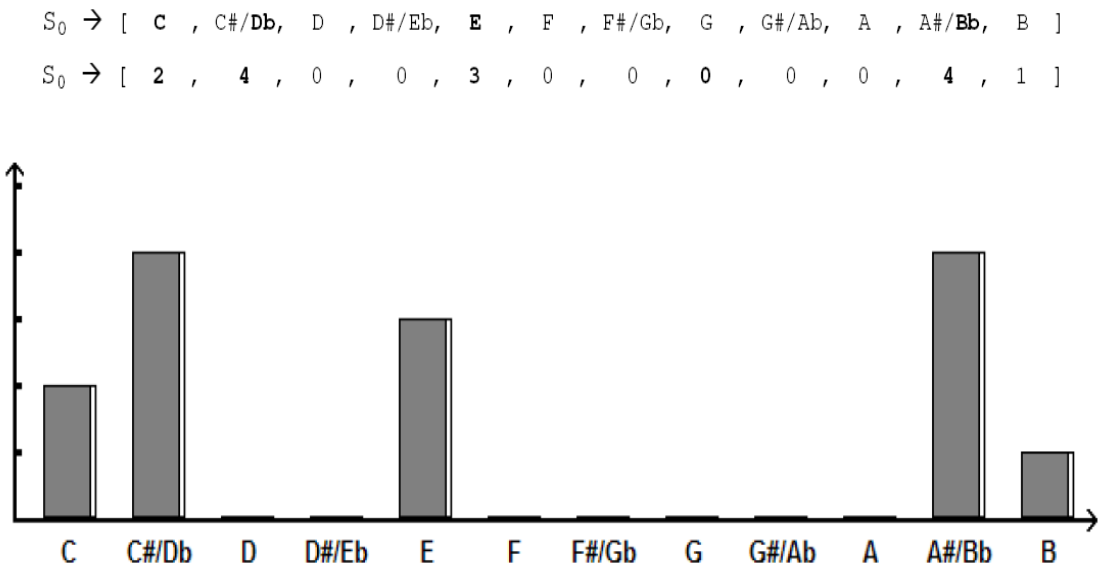

Figura 3.3 – Técnica de representação do vetor S0.

Scholz constatou que utilizando a abordagem original de Birmingham e Pardo sobre os dados não quantizados causava um particionamento excessivo. Ao invés de considerar ponto de partição todo elemento NOTE ON e NOTE OFF ele considerou apenas os eventos NOTE ON. Além disso, para ser considerado um ponto de partição válido é necessário que três ou mais ataques necessários no mesmo ponto, o num intervalo de tempo pré-definido. Este tempo foi definido com a duração de uma semi-colcheia a 100 BPM [Scholz, 2008].

Após essa fase de detecção dos segmentos, um pós-processamento é realizado. Neste pós-processamento é percorrida a lista de segmentos,

comparando-se cada par de segmentos consecutivo e unificando-os se os seus estados forem idênticos.

3.2.2 Escolha dos acordes mais prováveis

Os acordes mais prováveis são escolhidos a partir de uma lista de padrões que são comparados com os segmentos. Está lista foi criada de forma a incluir os acordes básicos e os mais freqüentes na harmonia jazzística. Além disso, foram mantidos somente os acordes que maximizassem a detecção destes, o que eu foi feito de forma empírica. Da comparação de cada padrão com o segmento é gerado um valor numérico de adaptação de cada acorde. A tabela 3.1 mostra a lista desses acordes.

Padrão	Descrição	Notação (Em C)
[0, 3, 7]	Acorde menor	Cm
[0, 3 , 6]	Acorde menor com quinta diminuta	$Cm^{(b5)}$
[0, 3, 6, 10]	Acorde menor com sétima e quinta diminuta (meio-diminuto)	$Cm^{7(b5)}$
[0, 3, 7, 10]	Acorde menor com sétima	Cm^{7}
[0, 1, 3, 10]	Acorde menor com sétima e nona menor	$Cm^{7(b9)}$
[0, 2, 3, 10]	Acorde menor com sétima e nona	$Cm^{7(9)}$
[0, 3, 7, 11]	Acorde menor com sétima maior	Cm^{7M}
[0, 2, 3, 11]	Acorde menor com sétima maior e nona	$Cm^{7M(9)}$
[0, 4, 7]	Acorde maior	C
[0, 4, 7, 10]	Acorde maior com sétima	C^{7}
[0, 2, 4, 10]	Acorde maior com sétima e nona	$C^{7(9)}$
[0, 2, 5, 10]	Acorde suspenso com sétima e nona	$C^{sus4(7)(9)}$
[0, 4, 8]	Acorde maior com décima terceira menor	$C^{(b13)}$
[0, 1, 4, 10]	Acorde maior com sétima e nona menor	$C^{7(b9)}$
[0, 1, 5, 10]	Acorde suspenso com sétima e nona menor	$C^{sus4(7)(b9)}$
[0, 4, 7, 11]	Acorde maior com sétima maior	C^{7M}
[0, 2, 4, 11]	Acorde maior com sétima maior e nona	$C^{7M(9)}$
[0, 3, 6, 9]	Acorde diminuto	C^{o}
[0, 4, 8, 10]	Acorde maior com sétima e décima terceira menor	$C^{7(b13)}$

Tabela 3.1 – Lista dos acordes mais usados na bossa nova.

Na notação cada elemento do vetor representa à distância em semitons da tônica considerada. Por exemplo, na tabela xx os acordes estão utilizando a tônica C(Dó), assim para o acorde maior temos a primeira nota com índice zero, que é a posição da própria nota C. A segunda nota tem índice 4, o que significa que é a nota C mais quatro semitons, portanto a nota E(Mi). E por último temos uma nota de índice sete, que significa sete semitons acima da nota C, o que equivale à nota G(sol).

Scholz utilizou a mesma função de utilidade proposta Birmingham e Pardo [Birmingham e Pardo, 2001 apud [Scholz, 2007]], para o cálculo do valor de adaptação de cada segmento ao padrão. A função atribui uma pontuação positiva se a nota existir no padrão e no segmento, negativa se houver no padrão e não houver no segmento e vice-versa(falhas). O quadro 3.1 mostra a função de utilidade.

Considere o estado e e o acorde c_{pt}, calculado a partir da tônica t aplicada ao padrão p. Além disso, considere E como a evidência positiva, N como a evidência negativa e F como o número de falhas. A função de utilidade é calculada como:

$$f_u(e, p, t) = \mathrm{E} - (\mathrm{N} + \mathrm{F})$$

Quadro 3.1 – Função de utilidade para cálculo do valor de adaptação de um padrão p, sobre uma tônica t, a um estado e de um segmento.

Aplicando-se a função de utilidade para cada segmento, escolhe-se a lista dos acordes mais prováveis para cada segmento, descartando os que a pontuação for inferior a oitenta e cinco por cento do valor do de maior pontuação, valor que foi obtido empiricamente por meio de diversos experimentos, sendo este o que mostrou o resultado mais eficiente. O algoritmo usado no cálculo da função de utilidade é mostrado no quadro 3.2.

```
E <= 0, N <= 0, F <= 0
e = [e0, ..., e11]
cpt = [n0, ... , nn]
para cada elemento ei de e
        encontrado <= falso
        para cada elemento nj de cpt
                se ei = nj então
                        encontrado <= verdadeiro
                        pare laço
                fim do se
        fim do para
        se encontrado = verdadeiro então
                E <= E + 1
        senão
                N <= N + 1
        fim do se
fim do para

para cada elemento ni de cpt
        encontrado <= falso
        para cada elemento ej de e
                se ni = ej então
                        encontrado <= verdadeiro
                        pare laço
                fim do se
        fim do para
        se encontrado = falso então
                F <= F + 1
        fim do se
fim do para
```

Quadro 4.2 – Pseudo-código para cálculo da evidência positiva E, da evidência negativa N e das falhas F, na adaptação de um estado *e* de um segmento, a um acorde c_{pt}, calculado a partir de um padrão *p* e uma tônica *t*.

Uma vez escolhidos os acordes mais prováveis, constrói-se um grafo direcionado G(V, E) e acíclico contendo k grupos distintos de vértices, onde cada grupo de vértices representa os acordes mais prováveis de cada segmento. Esses grupos são dispostos em camadas, sendo cada uma totalmente conectada à seguinte. O grafo está ilustrado na figura 3.4.

Tendo sido construído o grafo o problema agora será a de escolha do vértice de cada camada do grafo, que represente de forma mais fiel à informação harmônica. Este problema será abordado na próxima seção.

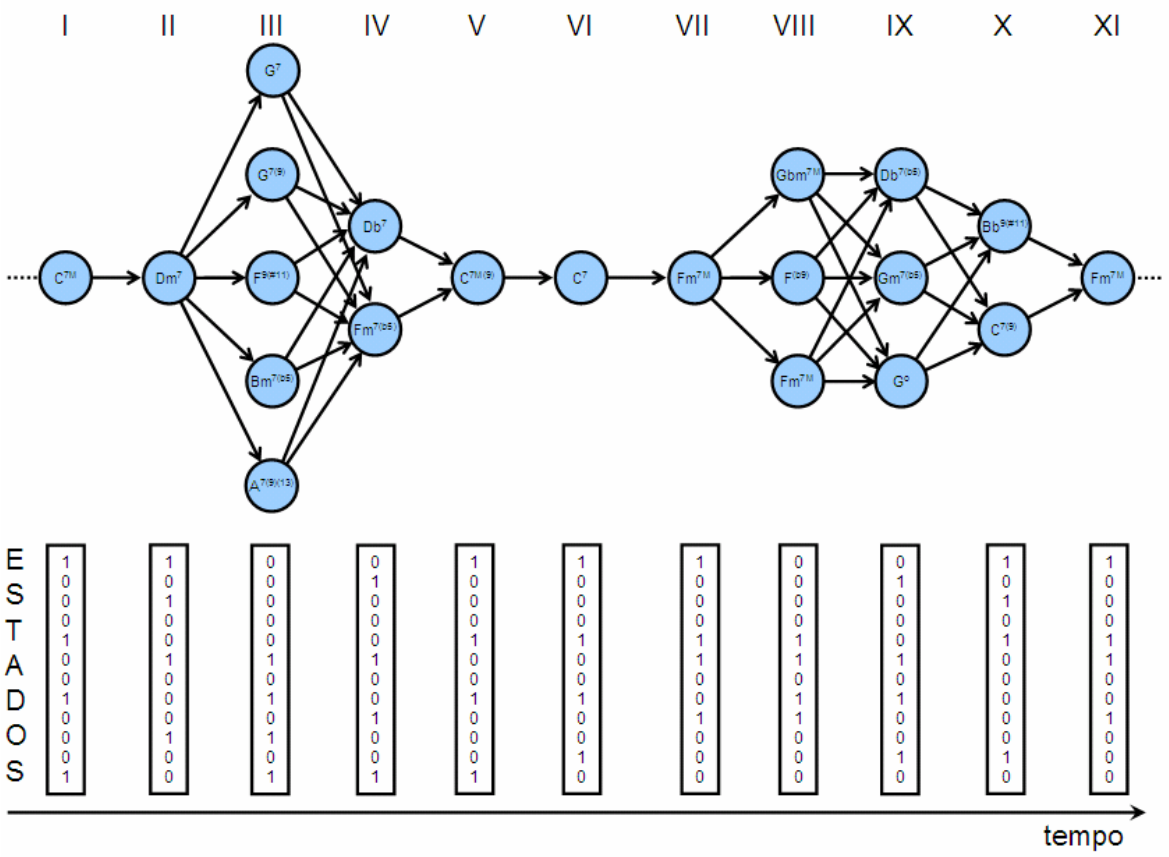

Figura 3.4 - Grafo construído a partir da segmentação inicial e escolha dos acordes com maior pontuação após a aplicação da função de utilidade sobre o estado de cada segmento e descarte dos acordes que não atingiram o limiar em cada segmento.

3.2.3 Análise Contextual para escolha do Melhor Acorde

Nesta etapa os grafos são subdivididos em subgrafos, onde cada subgrafo deverá iniciar e terminar com uma camada onde haja um acorde já identificado e no centro possuir pelo menos uma camada com acordes não identificados. Para se detectar qual o melhor acorde dentro de uma camada de um subgrafo são usadas regras usando o motor de inferência JEOPS [apud [Scholz, 2007]. Estas se baseiam nas seqüências de acordes mais recorrentes na harmonia jazzística e possuem ordem de prioridade. Se a regra de maior prioridade não é disparada, passa-se para regra seguinte e assim por diante. Uma vez que um acorde é escolhido por uma regra os demais da mesma camada são excluídos do grafo e os nós da camada são marcados como resolvidos. O processo é repetido com todos os subgrafos até não restar nenhum nó não resolvido.

3.3 Microtiming e Microdynamics

Na busca de entender melhor as sutilezas por trás da interpretação foram criadas as ferramentas do microtiming e microdynamics, para realizar uma análise mais profunda das modificações colocadas pelo intérprete quanto ao tempo e a dinâmica da música.

Para analisar os desvios de dinâmica e tempo na interpretação da bossa nova, primeiramente a música é dividida em frases, as quais na bossa nova correspondem a dois compassos. Numa frase cada compasso é dividido em dois intervalos, tempo 1 e tempo 2 para o primeiro compasso, tempo 3 e tempo 4 para o segundo compasso. Cada tempo então é subdividido em quatro intervalos de tempo iguais, utilizando para isso quatro marcadores chamados de time, semicolcheia, colcheia e semínima, situados respectivamente em um quarto, dois quartos, três quartos e no final da metade de cada compasso. Essa divisão de intervalos pode ser visto na figura 3.5.

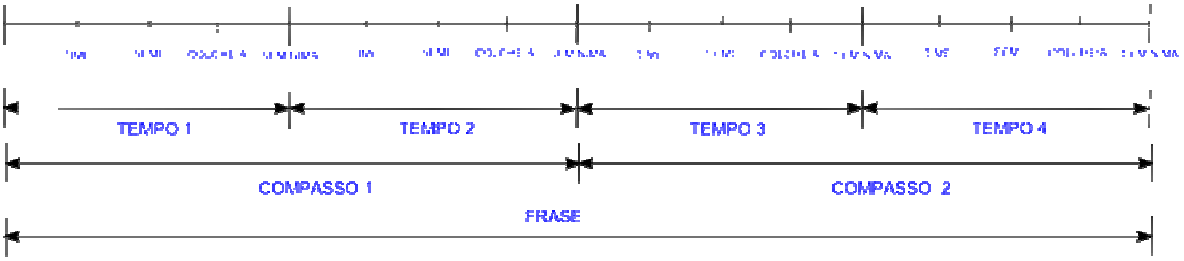

Figura 3.5 – A divisão de uma frase

Para uma análise dos desvios, cada nota executada deve se identificada como pertencente a um destes marcos, que será usado como referência para avaliação da nota. Mas aí surge um problema, como detectar a que marco cada nota pertence? Para resolver este problema foi determinado para cada marco uma janela de detecção, com um desvio correspondente a metade do tempo de um intervalo para a esquerda, e este mesmo intervalo para a direita. A nota pertencerá ao marco cuja janela a mesma esteja localizada. Na figura 3.6 ilustra as janelas e como as notas são identificadas.

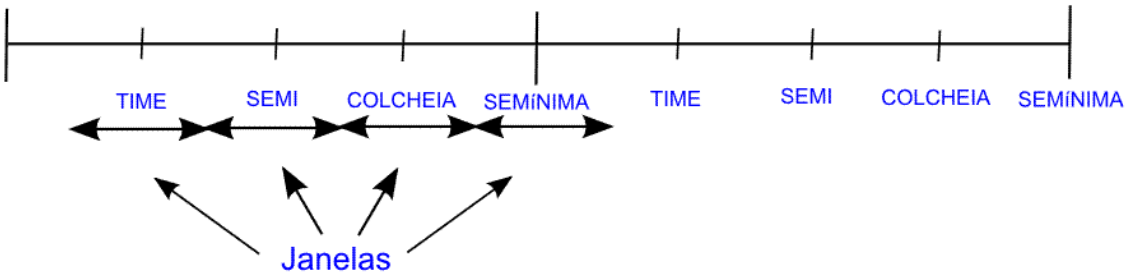

Figura 3.6 – Janelas de detecção das notas

Uma vez identificada a que janela pertence cada nota, e por conseqüência o marco, é calculado o valor do desvio de tempo da nota em relação ao seu marco, sendo positivo o desvio se a nota for executada à direita do marco e negativo se a nota for executada à esquerda do marco. Além disso, é armazenado o valor da dinâmica de cada nota relativa aquele marco.

De posse dos desvios de tempo e dos valores de dinâmica de cada nota são calculados as seguintes medidas:

- All Times – Calcula-se o valor médio dos desvios de tempo e de dinâmica de cada marco para todos os compassos.
- First/Second - Calcula-se o valor médio dos desvios de tempo e de dinâmica de cada marco para todos primeiros tempos de cada compasso, os sejam os tempos 1 e 3 de todas as frases (First) e para todos os segundos tempo de cada compasso, ou seja, para os tempo 2 e 4 de todas as frases (Second).
- Phrase – Calcula-se o valor médio dos desvios de tempo e de dinâmica de cada marco para todo cada tempos da frase ou seja os tempos 1, 2, 3 e 4.

4. Integração e Implementação

Neste capítulo descrevemos os passos necessários a integração, a modelagem e implementação do projeto. A seção 4.1 trata dos passos de integração, a seção 4.2 trata sobre a arquitetura do projeto, a seção 4.3 trata do suporte a Internacionalização e a seção 4.4 da interface gráfica.

4.1 Passos da Integração

O processo de integração foi dividido em quatro partes:

- Conhecimento individual dos subprojetos;
- União dos códigos dos subprojetos e retirada dos problemas;
- Criação de classes controladoras e criação de uma fachada comum a todos os subprojetos;
- Criação da interface gráfica unificada dos subprojetos.

A descrição de cada um desses passos será vista a seguir.

4.1.1 Conhecimento individual do funcionamento dos subprojetos

Neste passo foi realizado um estudo pormenorizado da arquitetura de cada subprojeto, estudando-se todos os pacotes, classes e suas relações. Foram extraídas informações sobre quais pacotes eram mais úteis, que classes poderiam ter uma função disponibilizada para o usuário, métodos que poderiam ser modificados para estáticos etc.

4.1.2 União dos Códigos dos Subprojetos e Retirada dos Erros

Neste passo os códigos individuais dos subprojetos foram unidos. Durante a união surgiram vários problemas que tiveram de ser resolvidos tais como: classes com o mesmo nome conflitando, classes que faziam a mesma

coisa com nomes diferentes, classes iguais com versões diferentes de código (mesmos métodos com códigos diferentes), livrarias faltando etc.

4.1.3 Criação das classes controladoras dentro dos subprojetos e criação da fachada

Uma vez isoladas as classes de maior interesse, foram separados os métodos que iriam ter uma aplicação direta com o usuário e para eles foram criadas as classes controladoras onde as mesmas estavam ausentes, sendo retirados códigos de outras classes que estavam acumulando a função de controlador juntamente com outra função. Por questão de modularidade, foi resolvido criar os controladores dentro dos pacotes já existentes para não dispersar o código. Em seguida foi modificada a fachada existente para a adição de novos métodos para interagir com os novos controladores criados.

4.1.4 Criação da interface gráfica unificada dos subprojetos

A interface gráfica foi construída de forma modular para facilitar uma expansão futura de novas ferramentas e dar suporte a Internacionalização. Os detalhes da Interface gráfica serão vistos na seção 4.4.

4.2 Arquitetura

4.2.1 Padrões de Projeto

Este projeto seguiu o modelo de projeto MVC (Model-View-Controller) [Fragmental, 2008]. O MVC faz uso de alguns design patterns já consagrados, como o Observer e o Strategy. Os componentes do MVC são divididos em três a saber:

- View – É a parte visível para o usuário, onde o mesmo seleciona as opções disponíveis e ou entra com dados, que pode ser uma interface gráfica (GUI), uma interface em modo texto (Command Line) ou uma interface web.

- Controller – Coordena as solicitações e informações vindas do usuário para sistema e as envia para a classe de destino no componente Model.
- Model – Representa o estado do sistema. Neste componente estão localizadas as classes de persistência.

As figuras 4.1 e 4.2 mostram dois diagramas de seqüência que representam respectivamente, o fluxo de informações do usuário para o sistema e do sistema para o usuário.

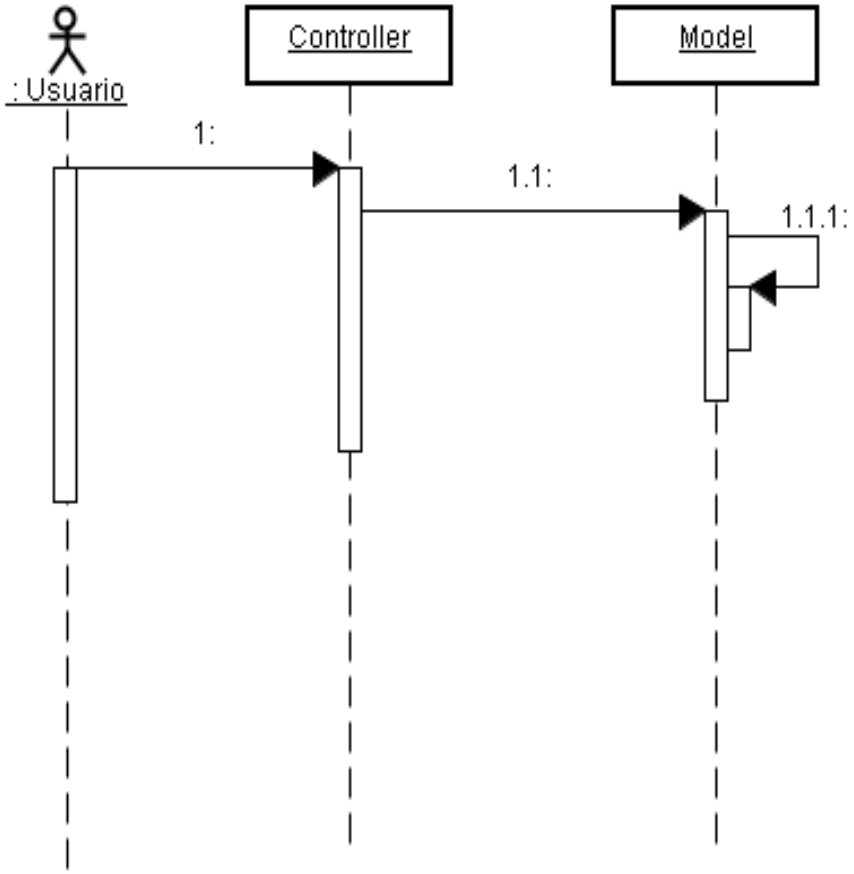

Figura 4.1 – Solicitação do usuário ao sistema. Fonte: [Fragmental, 2008]

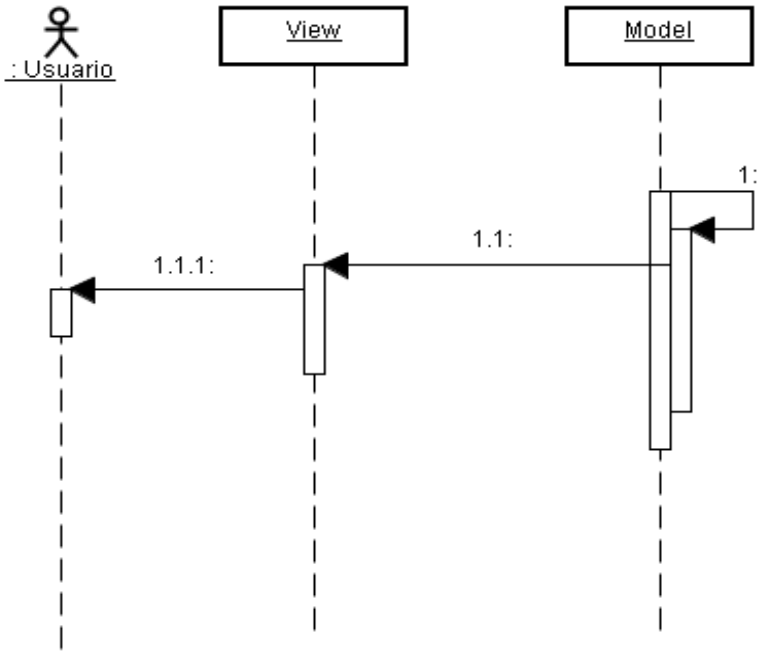

Figura 4.2 – Resposta do sistema ao usuário. Fonte: [Fragmental, 2008]

4.2.2 Composição da Arquitetura

A arquitetura é dividida em quatro componentes principais, as classes básicas, as classes controladoras, a fachada e a interface gráfica. As classes básicas são responsáveis pelo tratamento de baixo nível dado aos arquivos MIDI, e nelas são criadas abstrações para acordes, cifras etc. As classes controladoras são responsáveis por receber as solicitações oriundas do componente fachada e processá-las, utilizando-se para isso dos recursos das classes básicas. A fachada é responsável por receber as requisições provenientes da interface com o usuário e repassá-la para a classe controladora responsável pela tarefa, neste projeto criamos uma interface gráfica, mas nada impede que seja criada uma interface em modo texto. A interface gráfica é responsável por receber as solicitações do usuário, repassa-las para o componente fachada, mostrar os resultados na tela e tratar o erros e exceções que venham a ocorrer durante o processamento. A figura 4.3 ilustra a comunicação entre os componentes da arquitetura.

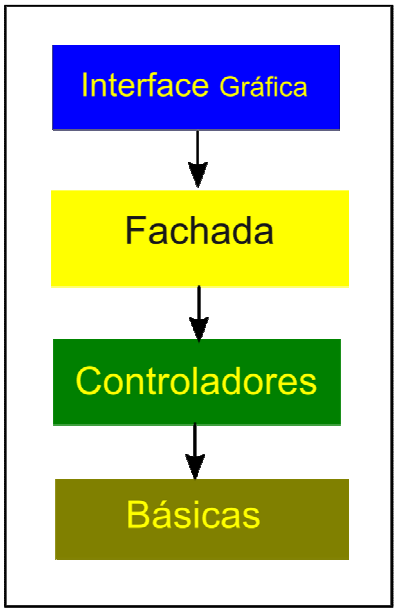

Figura 4.3 – Comunicação entre os componentes da arquitetura.

Nas seções a seguir veremos uma descrição das classes controladoras, que são as de maior importância.

4.2.2.1 Classes Controladoras

A seguir é dada uma descrição detalhada das classes controladoras.

StatisticsCalculatorControler - Responsável por calcular os resultados estatísticos da limpeza de arquivos MIDI capturados pelo violão e efetuada pelo framework interno[2] ou outro framework de limpeza, contra um arquivo que teve a limpeza efetuada por um especialista. Possui métodos para gerar a estatística para um único, ou múltiplos arquivos. Calcula as seguintes estatísticas: taxa de falsos negativos na deleção, taxa de falsos negativos na movimentação, taxa total de falsos negativos, taxa de falsos positivos na deleção, taxa de falsos positivos na movimentação e taxa total de falsos positivos. O método de gerar a estática de vários arquivos simultâneos não está implementado na interface gráfica. A figura 4.4 mostra a estrutura interna desta classe.

2

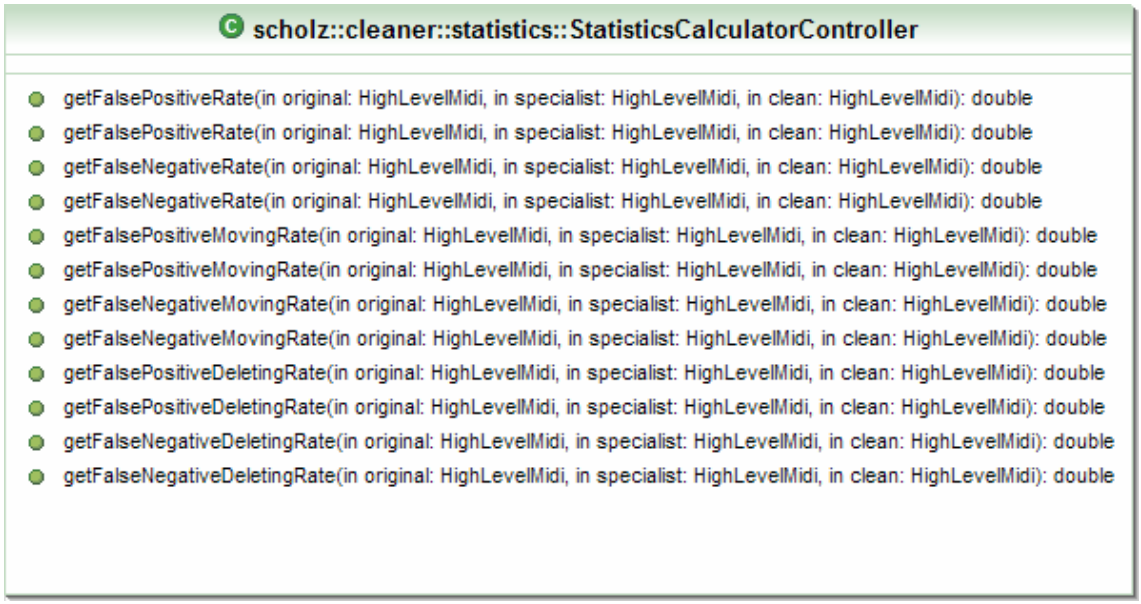

Figura 4.4 – Estrutura interna da classe StatisticsCalculatorControler

MidiCleanerControler - Responsável por efetuar a limpeza dos arquivos MIDI capturados pelo violão MIDI. Efetua a limpeza em duas fases: a primeira aplicando a base de regras simples, e a segunda aplicando a base de regras contextual, ambas descritas na seção 3.1. O método privado runFirstCleaningPhase é responsável pela execução da primeira fase e o método privado runHarmonyBasedCleaningPhase pela execução da segunda. Este controlador disponibiliza a opção de executar ou não a segunda fase. A primeira fase que utiliza a base de regras simples não é opcional, pois está é necessária a execução da segunda fase que aplica a base de regras contextual. A figura 4.5 mostra a estrutura interna desta classe.

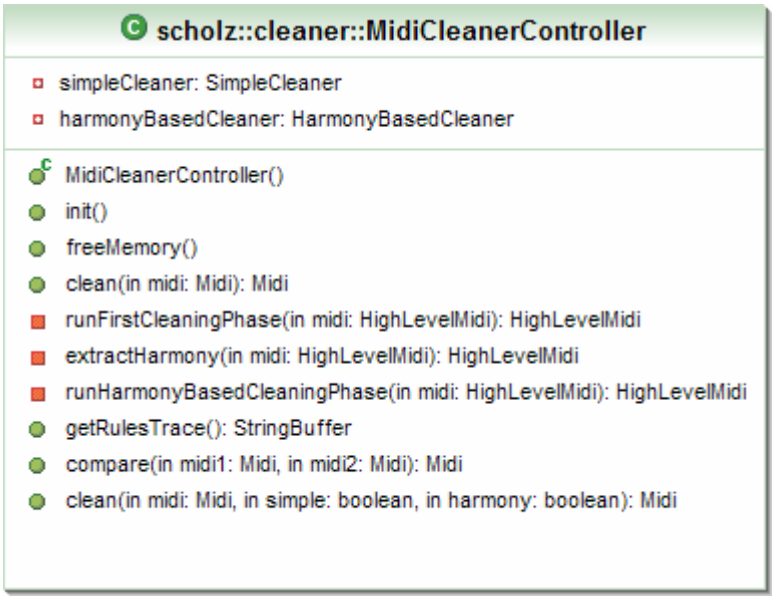

Figura 4.5 – Estrutura interna da classe MidiCleanerControler

HarmonyControler - Responsável por extrair as cifragem dos arquivos de cifra, gerar uma base de dados das cifras por estilo musical a partir de arquivos de cifra e extrair os acordes de um arquivo MIDI. Os métodos extractChords, extractCifra e extractCifraData, são respectivamente responsáveis pela extração dos acordes dos arquivos MIDI, pela extração da cifragem de um arquivo de cifra e pela geração da base de dados de cifras por estilo musical. A figura 4.6 mostra a estrutura interna desta classe.

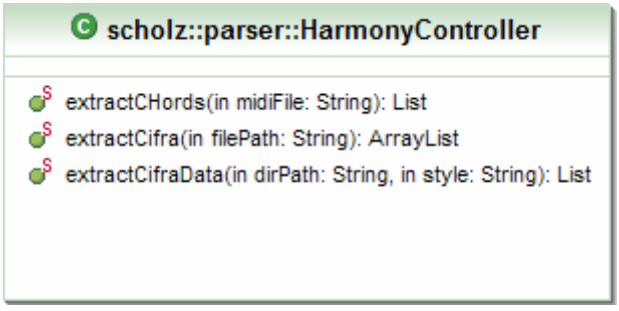

Figura 4.6 – Estrutura interna da classe HarmonyController

MicroAnalisysController - Responsável por analisar os arquivos MIDI localizados em uma pasta selecionada e calcular os valores de microtiming e microdynamics conforme foi descrito na seção xx. O método readFileDatas é o principal responsável pela extração das informações. Está classe é a junção dos dois controladores que existiam separadamente para calcular os valores de

microtiming e microdynamics. Tais controladores possuíam muito código em comum, por isso foram unificados. Um problema de design pattern encontrado nos controladores existentes foi que estes estavam chamando métodos da interface gráfica diretamente, uma violação ao padrão MVC, o que impossibilitava o reuso da fachada para criar um outro tipo de interface diferente da gráfica. A figura 4.7 mostra a estrutura interna desta classe.

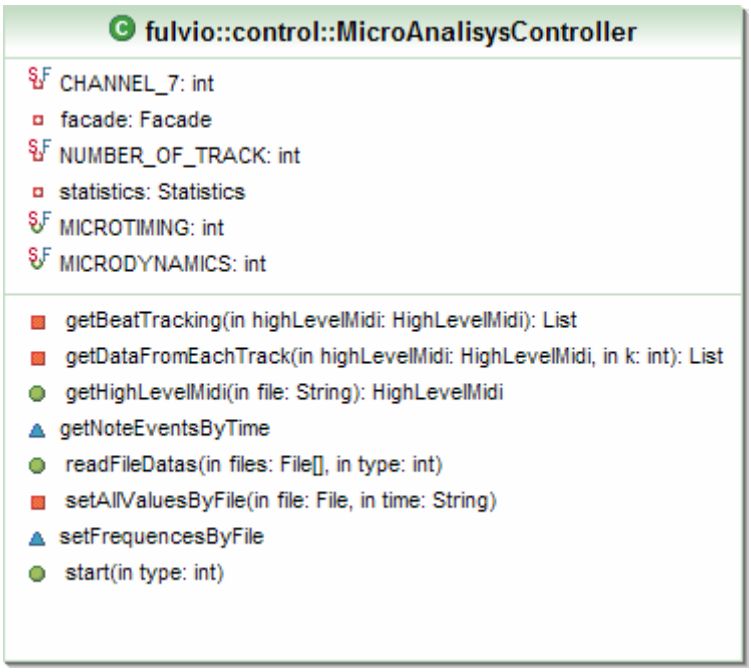

Figura 4.7 – Estrutura interna da classe MicroAnalisysController

4.3 Suporte a Internacionalização

O suporte a internacionalização foi implementado usando-se a classe Properties de Java, que lê e escreve arquivos de configuração contendo pares de textos do tipo *chave = valor*, para controlar as opções de idiomas são utilizados os arquivos *default.properties* e *config.properties*. No arquivo config.properties estão as configurações do sistema ao ser iniciado, tais como as opções de idiomas, o idioma padrão e o último diretório acessado. Caso esse arquivo não exista ou as opções estejam em branco, as opções serão carregadas do arquivo *default.properties*.

A seleção do idioma é controlada pelo valor do atributo *language*, cujo valor segue a sintaxe (sigla do idioma)_(sigla do pais), onde a sigla do idioma

segue o padrão ISO 639 de duas letras, e a sigla do país segue o padrão de representação de duas letras do ISO 3166.

Os arquivos com os textos de cada idioma estão localizados no pacote *locale* e são nomeados seguindo a sintaxe *language_(sigla do idioma)_(sigla do país)*, sendo as siglas as mesmas citadas no parágrafo anterior. Nestes arquivos, os textos seguem a sintaxe palavra-chave=valor. A palavra-chave representa um código para o texto que será apresentado na interface gráfica e que é comum em todos os idiomas, por exemplo, o texto “Abrir” em português do Brasil é registrado no arquivo Language_pt_BR da forma open=abrir e em inglês dos Estados Unidos no arquivo Language_en_US da forma open=Open. Assim para se adicionar suporte para uma nova língua se faz necessária a criação de um arquivo no pacote *locale* com o nome no formato descrito anteriormente, e copiar as palavras chave de outro arquivo já existente e traduzi-las para o novo idioma, além de adicionar o novo idioma na classe MyMenu.

4.4 Interface Gráfica

A interface gráfica é composta por um menu com as opções de configuração e as várias ferramentas disponíveis por meio de um painel de abas. A classe principal é a *Gui,* que é uma composição das classes MyMenu e SelectTools. A classe MyMenu agrupa as opções de configuração incluindo a seleção do idioma da interface gráfica e a classe SelectTools é utilizada para fazer o gerenciamento das ferramentas nas abas. Optamos por criar um painel separado para cada ferramenta dentro da classe SelectTools, para desta forma diminuir a complexidade que haveria se fosse uma classe única, aumentar a modularidade e consequentemente facilitar a inclusão de novas ferramentas na aplicação. A figura 4.8 ilustra as partes principais da interface gráfica.

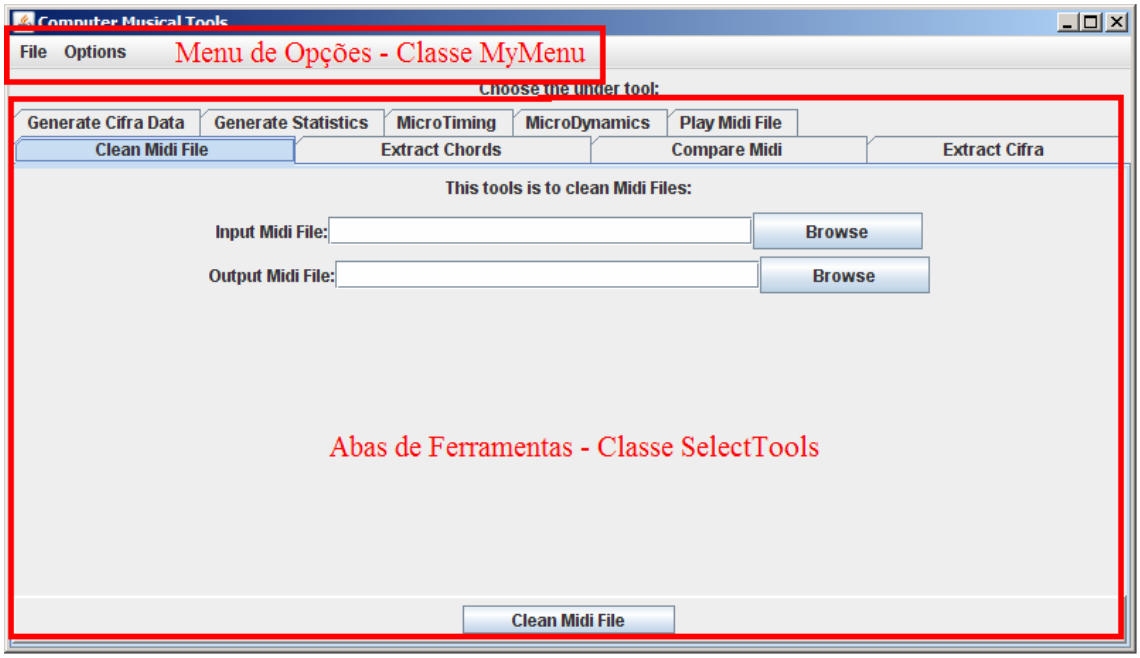

Figura 4.8 – Partes principais da interface gráfica

Cada painel na classe SelectTools é uma classe filha da classe JPanelModel, que é composta de três campos: descrição, controles de entrada e botões de ação. O campo descrição é utilizado para descrever a ferramenta, e é passada no construtor da classe JPanelModel. O campo controles de entrada é onde ficam localizados os campos de entrada do usuário com a interface, que pode ser, por exemplo, a localização de um arquivo de entrada, de um arquivo de saída, de um diretório etc. Já no campo botões de ação é onde ficam localizados os botões que irão agir sobre os dados passados pelo usuário nos campos de controle de entrada. A composição de um painel filho da classe JpanelModel pode ser observada na figura 4.9.

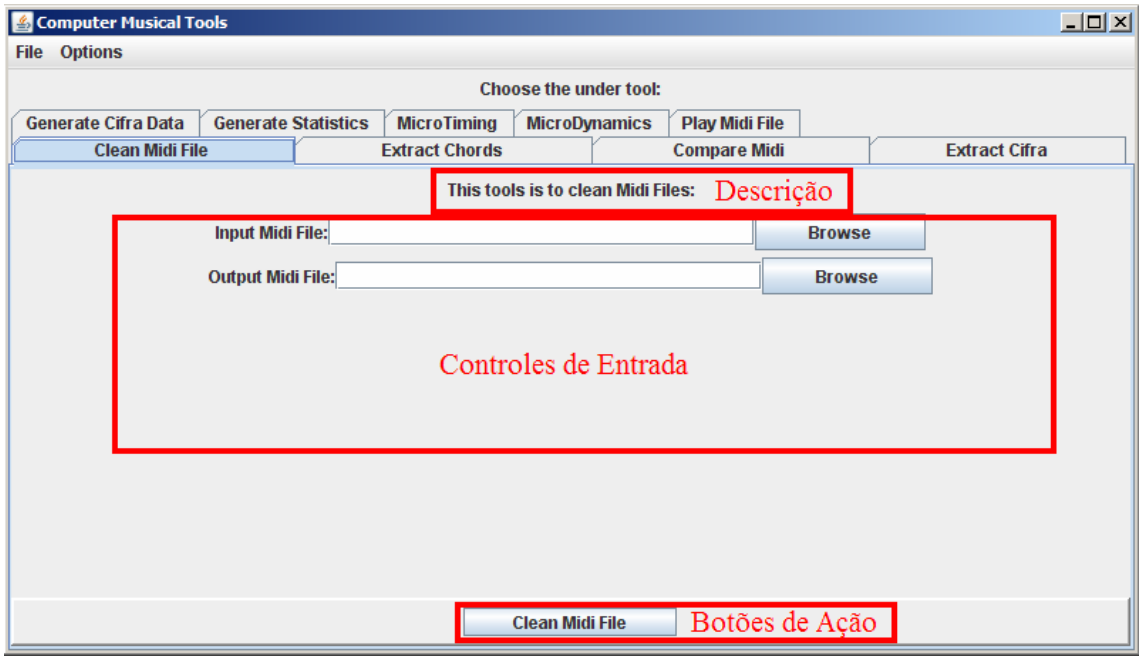

Figura 4.9 – Partes de uma classe JPanelModel

Para facilitar a busca de informações no sistema de arquivos, tais como o caminho de arquivo ou de um diretório, algo muito comum em todas as ferramentas, foi criada a classe InputOutputPanel especializada nesta tarefa. A classe InputOutputPanel possui em sua interface a descrição do arquivo a ser selecionado, uma caixa de texto contendo o caminho deste arquivo, um botão de navegação e um botão opcional de ação sobre o próprio arquivo. Além disso, encapsula todas as funções necessárias para navegação e seleção de um arquivo no sistema, incluindo filtros para navegação em arquivos MIDI, arquivos texto e diretórios. A figura 4.10 ilustra a aparência da classe InputOutputPanel na interface gráfica.

Figura 4.10 – Aparência da classe InputOutputPanel

4.4.2 Ferramentas da interface gráfica

Nesta seção descreveremos a interface das ferramentas deste projeto, com uma descrição detalhada das interfaces com controles mais complexos.

4.4.2.1 Limpar Arquivo MIDI

Esta interface é a responsável pela limpeza de um arquivo MIDI. A figura 4.9 mostra esta interface. O primeiro botão navegar é usado para o usuário selecionar o arquivo MIDI de entrada onde será efetuada a limpeza. O segundo botão é usado para o usuário selecionar onde deverá ser salvo arquivo limpo e por fim o botão limpar arquivo MIDI dá início ao processo de limpeza.

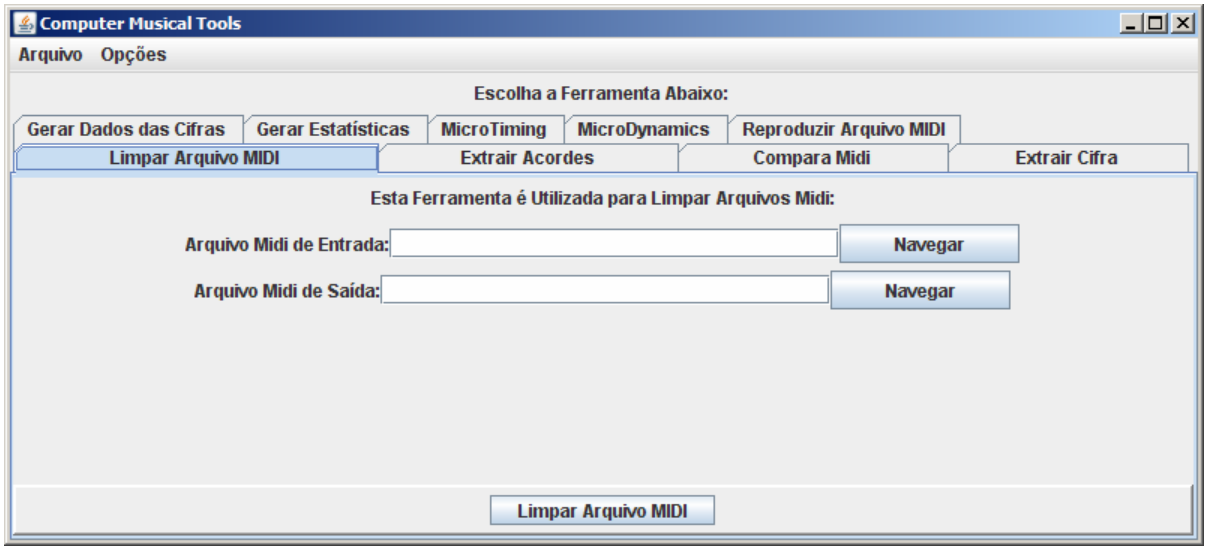

Figura 4.9 – interface da ferramenta Limpar Arquivo MIDI

4.4.2.2 Extrair Acordes

Esta interface é responsável pela extração dos acordes de um arquivo MIDI. A interface é composta pelo botão navegar que é responsável pela seleção do arquivo MIDI, por uma tabela que exibe os acordes extraídos, pelo botão iniciar que dá início ao processo de extração e pelo botão Exportar que exporta os dados da tabela para uma planilha eletrônica. A figura 4.10 mostra esta interface.

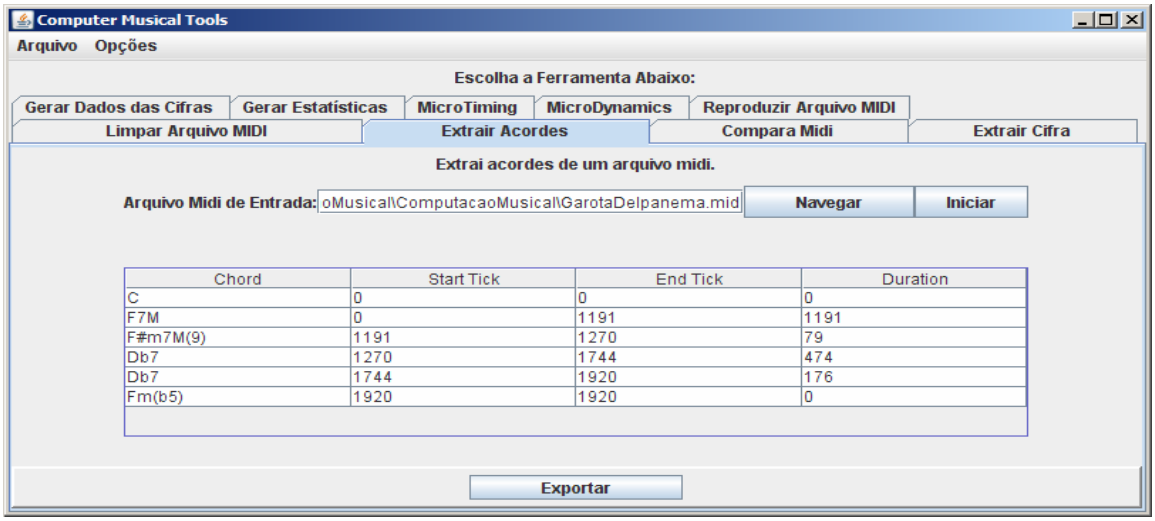

Figura 4.10 – interface da ferramenta Extrair Acordes

4.4.2.3 Compara MIDI

Esta interface é responsável pela geração de um arquivo MIDI de saída que é o resultado da comparação de dois arquivos MIDI de entrada. Na primeira trilha do arquivo de saída estão as notas comuns aos dois arquivos de entrada, na segunda trilha as notas que estão no primeiro arquivo mais não no segundo e na terceira trilha as notas que estão no segundo arquivo mais não no primeiro. A figura 4.11 mostra esta interface.

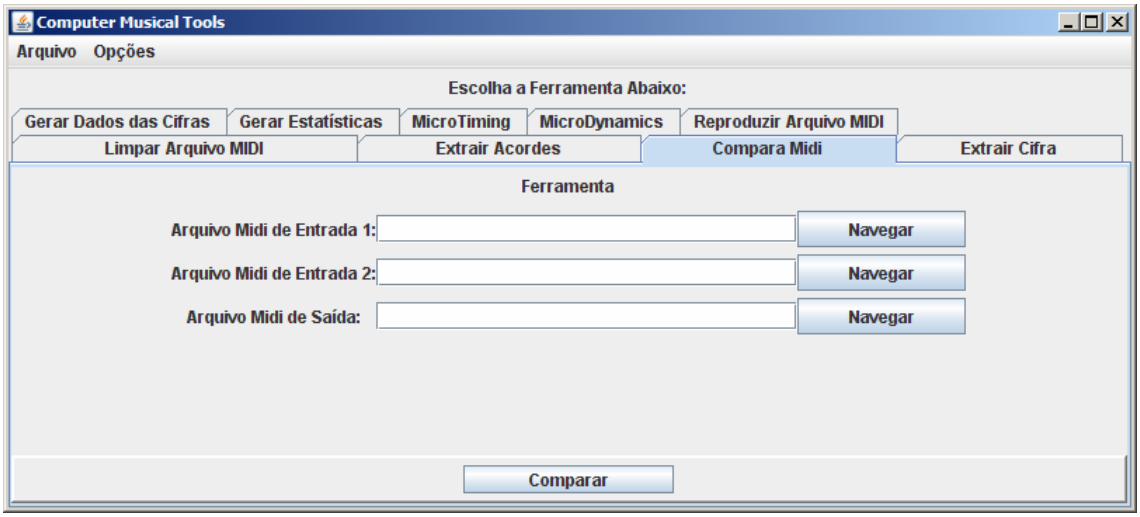

Figura 4.11 – interface da ferramenta Compara MIDI

4.4.2.4 Extrair Cifra

Esta interface é responsável pela extração das cifras de um arquivo texto contendo a letra e as cifras de uma música. As cifras extraídas são mostradas linha a linha em uma caixa de texto, cujo conteúdo pode ser exportado para um arquivo. A figura 4.12 mostra esta interface.

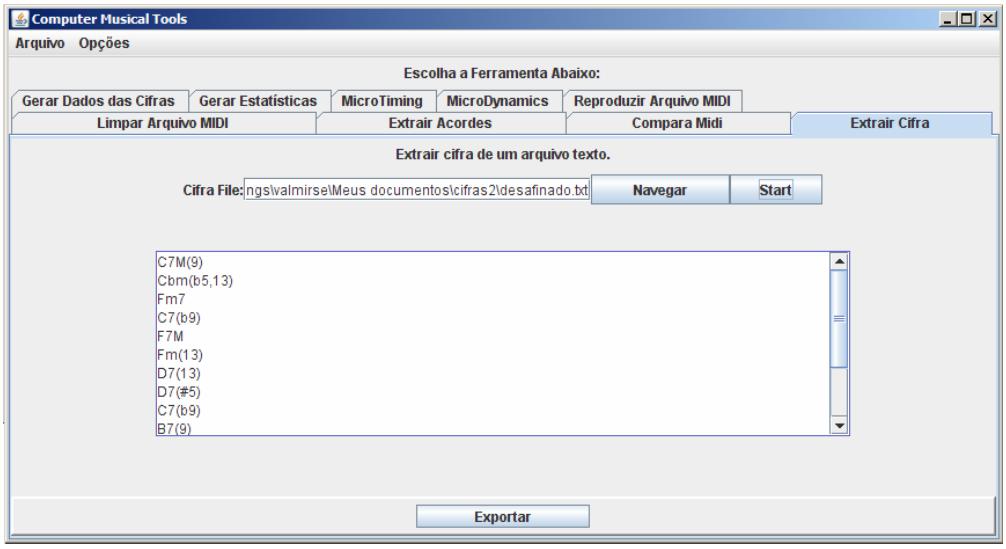

Figura 4.12 – interface da ferramenta Extrair Cifra

4.4.2.5 Gerar dados das Cifras

Esta interface é responsável por gerar uma base de dados de cifras por estilo musical a partir de arquivos de cifra de entrada. Os dados obtidos podem ser exportados para um arquivo texto. A figura 4.13 mostra está interface na língua inglesa.

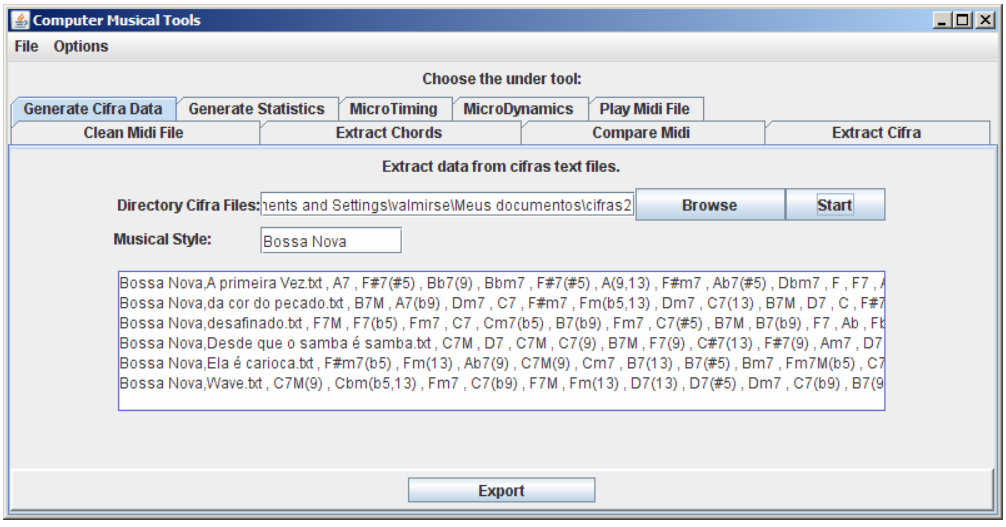

Figura 4.13 – Interface da ferramenta Gerar dados das Cifras

4.4.2.6 Gerar Estatísticas

Esta interface é responsável por gerar o resultado estatístico da limpeza dos arquivos MIDI por meio do controlador **StatisticsCalculatorControler.** Para calcular os resultados é necessário selecionar os arquivos de entrada, selecionar os tipos de estatística que deverão ser calculados e pressionar o botão Gerar Estatísticas. Selecionando-se outro conjunto de entrada e pressionando novamente *Gerar Estatísticas* uma nova linha de resultado é gerada na tabela. Desejando-se reiniciar os cálculos com outros arquivos tabela basta pressionar o botão *Limpar Tabela*. O botão *Export* serve para exportar os dados estatísticos da tabela para uma planilha eletrônica. A figura 4.14 mostra está interface na língua inglesa.

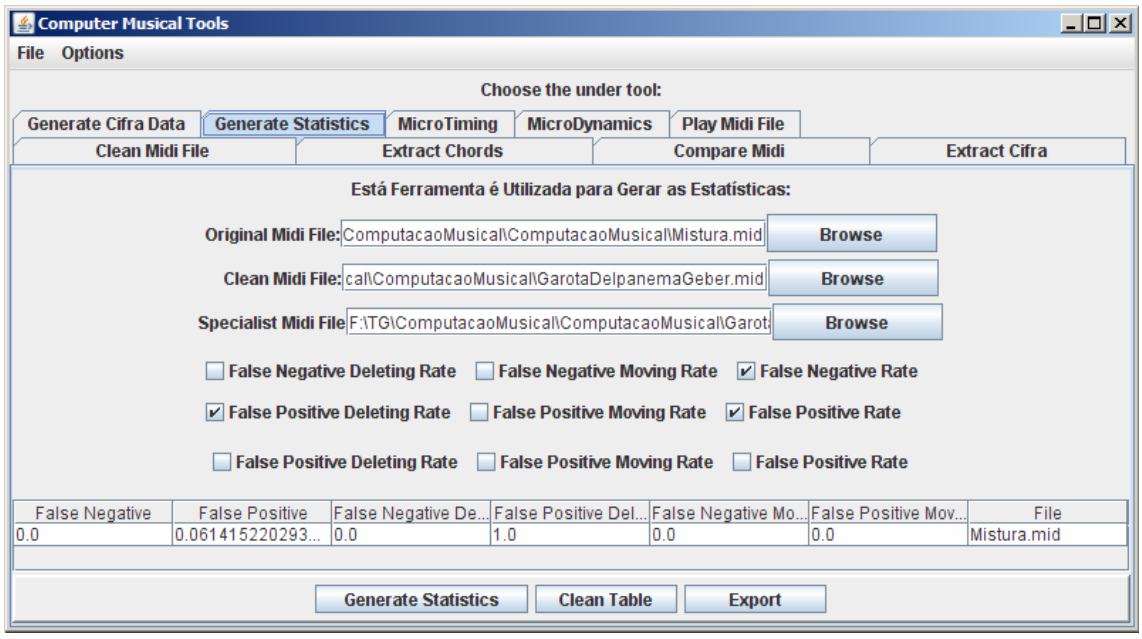

Figura 4.14 – Interface da ferramenta Gerar Estatísticas

4.4.2.7 Microtiming e MicroDynamics

Estão duas ferramentas de interfaces idênticas calculam os valores de microtiming e microdynamcs vistos na seção xx. Para calcular os valores deve-se selecionar o diretório onde estão os arquivos MIDI e pressionar o botão *Iniciar*. Os resultados podem ser exportados para uma planilha do excel. A figura 4.15 mostra a interface de microdynamics na língua inglesa.

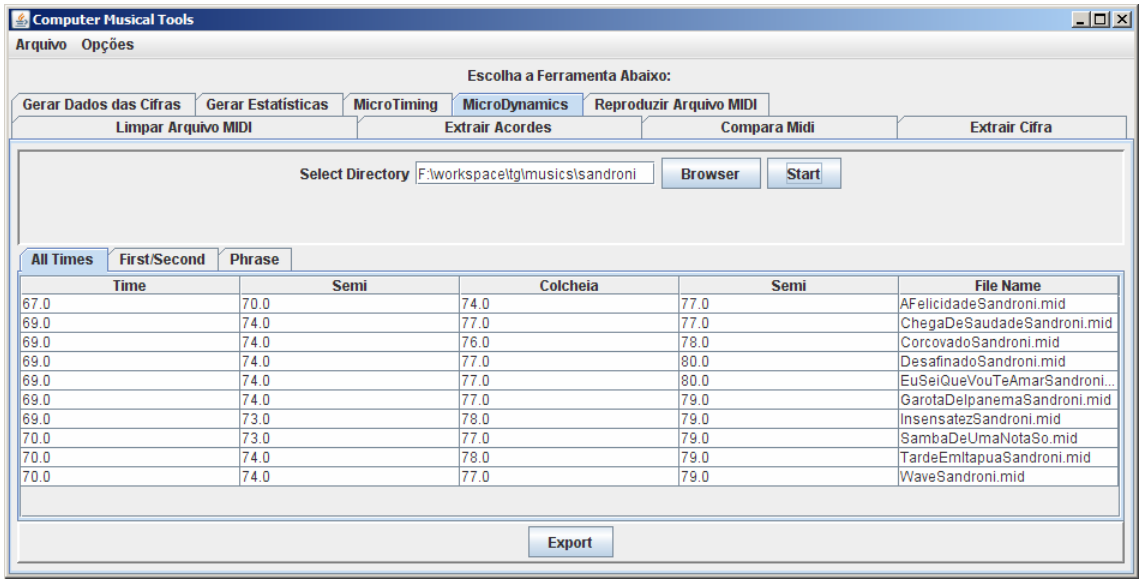

Figura 4.15 – Interface das ferramentas Microtiming e MicroDynamics

4.4.2.8 Reproduzir Arquivo MIDI

Esta interface é usada para se reproduzir um arquivo MIDI selecionado. A figura 4.16 mostra a interface na língua inglesa.

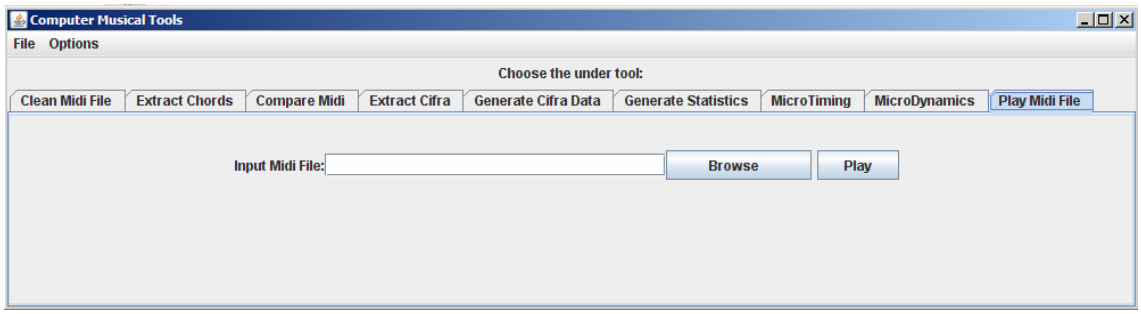

Figura 4.16 – Interface da ferramenta Reproduzir Arquivo MIDI

4.4.2.9 Extrair Padrão Rítmico

Está interface é utilizada para chamar o aplicativo criado por Trajano para extrair padrões rítmicos em um arquivo MIDI [Trajano, 2007]. Para se utilizar a ferramenta deve-se selecionar o arquivo pressionando o botão browse e em seguida pressionar o botão start. A figura 4.17 mostra a interface na língua inglesa.

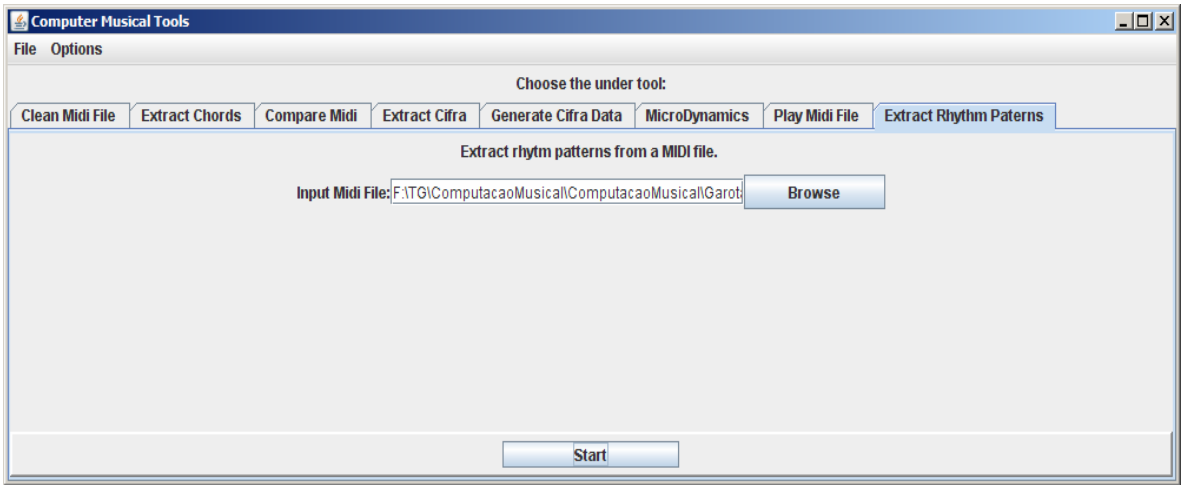

Figura 4.17 – Interface da Ferramenta Extrair Padrão Rítmico

5. Conclusão e Trabalhos Futuros

Este trabalho integrou as diversas ferramentas desenvolvidas no Centro de Informática da Universidade federal de Pernambuco, na área de computação musical, criando para elas uma interface gráfica amigável, de fácil utilização, e de forma a permitir uma fácil integração de novas ferramentas. Criou uma ferramenta para extração de cifras de arquivos texto para geração de uma base de dados e acrescentou também uma ferramenta de reprodução de arquivos MIDI. Infelizmente pela escassez de tempo, pelo recebimento tardio do material relativo à extração dos padrões rítmicos, e de sua alta complexidade, não foi possível criar a ferramenta de extração de padrões dos padrões rítmicos.

Assim foi desenvolvido um aplicativo único e certamente um dos mais completos para auxiliar a pesquisa na área de computação musical, no que concerne as áreas de interpretação musical utilizando-se do violão brasileiro para análise de ritmos como bossa nova e MPB.

Em trabalhos futuros seria importante a criação da ferramenta de extração de padrões de padrões rítmicos, uma melhor integração entre os trabalhos desenvolvidos por Scholz e Trajano retirando-se os códigos redundantes, e a adição de suporte para novos idiomas na interface além do Inglês e do Português existentes.

6. Bibliografia

Friberg, A. 2008. The Science of Music Performance. Disponível em http://www.speech.kth.se/music/performance/. Acessado em 02 de outubro de 2008.

Liu at al, 2005. An Efficient Approach to Extracting Approximate Repeating Patterns in Music Databases. International Conference on Database Systems for Advanced Applications 2005

Liu at al, 2004. Finding Approximate Repeating Patterns from Sequence Data. International Symposium on Music Information Retrieval, 2004.

Musical performance. In: Enciclopédia Britânica On-line 2008. Disponível em http://www.britannica.com/EBchecked/topic/399251/musical-performance. Acessado em 20 de Setembro de 2008.

MVC e Camadas. Disponível em http://www.fragmental.com.br/wiki/index.php?title=MVC_e_Camadas. Acessado em 05 de Setembro de 2008.

Scholz, Ricardo. 2008. COCHONUT: um processo para reconhecimento de acordes em seqüência capturadas por violões MIDI. Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Pernambuco 2008.

Trajano, Ernesto. 2007. Descoberta automática de conhecimento em interpretações musicais: O caso do acompanhamento rítmico ao violão. Tese (Doutorado em Ciência da Computação) – UFPE 2007.

Witten, Ian H; Frank , Eibe. Data mining: practical machine learning tools and techniques – segunda edição. Morgan Kaufmann 2004.