Adriano de Oliveira Caminha

MHITS - Um Sistema Tutor Inteligente em Harmonia Musical

Campina Grande
2000

MHITS - Um Sistema Tutor Inteligente em Harmonia Musical

Adriano de Oliveira Caminha

Dissertação apresentada ao Curso de Mestrado em Informática da Universidade Federal da Paraíba, como exigência parcial para a obtenção de Grau de Mestre.

Área de Concentração: Ciência da Computação

Orientadores:

Prof. Edilson Ferneda

Prof. Evandro de Barros Costa

Campina Grande

Ficha catalográfica

Caminha, A. de O. – MHITS, Um Sistema Tutor Inteligente em Harmonia Musical. Campina Grande, UFPB/CCT, 2000.
131p.

1. Computação e Música. 2. Informática na Educação. 3. Educação a Distância. I Título.

CDD 001.642

MHITS – UM SISTEMA TUTOR INTELIGENTE EM HARMONIA MUSICAL

ADRIANO DE OLIVEIRA CAMINHA

DISSERTAÇÃO APROVADA EM 29.02.2000

PROF. EDILSON FERNEDA, Dr.
Orientador

PROF. EVANDRO DE BARROS COSTA, Dr.
Orientador

PROF. BERNARDO LULA JÚNIOR, Dr.
Examinador

PROF. ARTURO HERNÁNDEZ DOMÍNGUEZ, Dr.
Examinador

LUCIÊNIO DE MACÊDO TEIXEIRA, M.Sc
Examinador

CAMPINA GRANDE – PB

Este trabalho é dedicado

ao meu pai, José,
à minha mãe, Neide,
à minha amada, Vera

Agradecimentos

A Deus.

Aos meus pais, José e Neide, pelo amor, ensinamentos, paciência e dedicação.

Ao meu orientador e amigo, Edilson Ferneda, por me guiar durante este trabalho, pela amizade e por todo o apoio desde a graduação.

À minha amada, Vera, pelo amor, pelos conselhos, por sua intuição magnífica, pela paz e alegria que transmite a todos, pelo companheirismo, pela compreensão e paciência, por tudo.

À Evandro Costa e Luciênio Teixeira pelas discussões, conversas e contribuições para este trabalho.

Aos professores Bernardo Lula, Arturo Dominguez pelas contribuições e amizade.

Aos meus irmãos Leonardo, Thaísa, Sabrina e Alessandro. Aos também irmãos, Arthur e Jordana.

Aos professores Mário Benevides e Inês C. Dutra pela acolhida na COPE/UFRJ e por todo o apoio e amizade.

À Rosa Costa pela amizade e apoio imprescindíveis.

À professora Socorro Paz, pela amizade, paciência e orientação antes mesmo de iniciar este trabalho.

Aos meus irmãos de banda, Zé Carlos, Alessandro e Francisco, pelo companheirismo, pela amizade e pelos momentos maravilhosos em que tocamos juntos.

Ao meu amado cão Wolfgang (in memoriam) pela segurança, pelas noites em que esquentava os meus pés enquanto eu trabalhava, pela alegria e fidelidade. Perdão por não ter te dado toda a atenção e dedicação que você merecia.

Aos meus amigos e colegas de turma, pelo companheirismo, pela amizade, pelos momentos de diversão.

Aos amigos Gilvan, Rodrigo, Cremildo, Lurdes, Cláudia, Solange, Sueli, Mercedes, Lúcia, Marli, Sônia, Carlos, Daniele, Vânia "Loira" e Vânia "Morena", Douglas e todos os que conheci no Rio de Janeiro, pela força e amizade.

À Aninha e Vera, pela paciência, amizade, dedicação e atenção. À Inês, pelas tapiocas com queijo maravilhosas. À Manuela, Zeneide, Teresinha, Lili e Agda, pela amizade e apoio.

A todos os amigos que não foram aqui citados.

Resumo

Este trabalho propõe um ambiente baseado em agentes, o MHITS – Sistema Tutor Inteligente em Harmonia Musical. Nele são abordados aspectos da IA, da Computação e Música, da Representação de Conhecimento em Harmonia Musical, do desenvolvimento de Sistemas Tutores Inteligentes, todos estes aplicados ao desenvolvimento de software educacional direcionado à utilização em aulas presenciais ou a distância, com o auxílio de especialistas humanos. Focalizando o aprendizado de Harmonia Musical, dando continuidade aos trabalhos iniciados com a representação do conhecimento neste domínio [TEIX 97] e utilizando o modelo MATHEMA [COST 97b], o MHITS contempla uma interface hipermídia multiplataforma, com características de acesso via Internet (WWW), incluindo um mecanismo de comunicação entre applets JAVA e aplicações desenvolvidas em Prolog.

Abstract

This work proposes an agents based environment, the MHITS – Musical Harmony Intelligent Tutoring System. It aproaches AI aspects, Computer Music, Musical Harmony Knowledge Representation, development of Intelligent Tutoring Systems, all of them applied to educational software development to be used either on present learning or distance learning, with the aid of human specialists. Focusing on the learning of Musical Harmony, giving a continuity to a previous work in knowledge representation in this domain [TEIX 97] and based on MATHEMA environment [COST97b], the MHITS work through a platform-independent hypermedia interface, with performance via Internet (WWW), including a communication component between JAVA applets and Prolog based applications.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1
Introdução

A Inteligência Artificial (IA) desenvolveu-se subdividida em várias subáreas. Dentre elas, uma que vem se destacando é a dos Sistemas Baseados em Conhecimento (SBC), sistemas que usam o conhecimento sobre algum domínio para gerar seus resultados. Estes sistemas possuem como característica principal a separação entre o conhecimento do domínio do problema e o conhecimento geral da solução do problema.

Este trabalho aborda uma subárea dos SBC que vem se desenvolvendo bastante e se mostrando adequada à finalidades educativas, os Sistemas Tutores Inteligentes (STIs), que integram técnicas de IA e uma teoria psicológica de aquisição de conhecimento dentro de um plano de ensino, oferecendo ambiente instrucional individualizado apoiado por computador. [COST 98b]

Educação (ou mais especificamente, instrução) é uma das áreas onde as aplicações de IA mais têm crescido nos últimos tempos. A pesquisa em STIs ou em Sistemas Inteligentes de Instrução Assistida por Computador (ICAI) é exemplo concreto dessa interação. [VICC 92]

Por outro lado, pesquisadores de diversas áreas do conhecimento têm se interessado cada vez mais pelo domínio da Música. A Computação, em particular, tem encontrado neste domínio um campo bastante interessante não só para a validação de resultados existentes, mas também para contribuir para a resolução de problemas específicos e intrinsecamente ligados ao domínio da Música. [TEIX 97]

A área de Computação e Música envolve normalmente um investimento em linhas tais como: Análise Musical, Composição e Ambientes de Apoio ao Ensino-aprendizagem. Neste sentido, o presente trabalho se situa nesta área, mais especificamente na concepção e desenvolvimento de um Sistema Tutor Inteligente em Harmonia Musical.

Esse trabalho é de natureza interdisciplinar, envolvendo aspectos de Inteligência Artificial (por exemplo: Representação de Conhecimento) e de Educação, além do domínio da

Música (Harmonia).

1.1. Motivação

Este trabalho tem como motivação principal a implementação de um Sistema Tutor Inteligente em Harmonia Musical, tendo como base os trabalhos iniciados em [TEIX 97] e [COST 97], onde temos um esboço da representação imprescindível para representar os aspectos de Harmonia que serão utilizados, bem como o modelo MATHEMA, que foi utilizado para a formalização de algumas partes desse trabalho, como na representação de conhecimento baseada em agentes inteligentes.

1.2. Objetivos

O objetivo principal é a modelagem e implementação de um STI em Harmonia Musical denominado MHITS – *Musical Harmony Intelligent Tutoring System*, para o auxílio no ensino e aprendizagem deste domínio.

Como objetivos secundários, a capacitação na área de conhecimento, bem como obter-se um retorno experimental sobre o desenvolvimento de um STI em todos os aspetos envolvidos (modelagem do sistema, desenvolvimento da interface, integração de recursos multimídia).

1.3. Organização do trabalho

Este trabalho está organizado, além desta introdução, em mais cinco capítulos:

Capítulo II – *Sistemas Tutores Inteligentes* - Iniciaremos fundamentando os princípios básicos dos STIs. Após um breve histórico que abrange desde o surgimento dos sistemas de instrução auxiliada pelo computador até o surgimento dos STIs, veremos algumas arquiteturas desenvolvidas por vários pesquisadores da área, descreveremos os componentes básicos da estrutura de um STI, estudaremos três STIs baseados em arquiteturas multi-agentes, a saber: o Expert Piano, o SEI - Sistema de Ensino Inteligente e o MATHEMA, bem como traremos à tona algumas questões concernentes ao desenvolvimento destes sistemas.

Capítulo III – *O MATHEMA* – Discutiremos, neste capítulo, o ambiente MATHEMA, o modelo de representação do conhecimento deste ambiente e o esboço de uma representação da Harmonia Musical segundo os moldes do MATHEMA desenvolvido em [TEIX 97].

Capítulo IV – *Harmonia Musical* - Serão discutidos neste capítulo, os aspectos fundamentais da Harmonia bem como quais aspectos serão enfocados neste trabalho. Apresentaremos ainda, um esboço para exercícios em Harmonia Musical, que servirão como ponto de partida para o desenvolvimento do nosso tutorial.

Capítulo V – *O Sistema Tutor Inteligente em Harmonia Musical* – Neste capítulo teremos o objeto de pesquisa deste trabalho, ou seja, a modelagem e desenvolvimento do STI em Harmonia Musical, incluindo aspectos importantes para o entendimento do funcionamento deste.

Capítulo VI – *Conclusões e Trabalhos Futuros* – Chegaremos então ao final do trabalho com as conclusões e contribuições, além de apresentarmos sugestões para trabalhos futuros.

Capítulo 2
Sistemas Tutores Inteligentes

2.1. Introdução

A informática destaca-se entre as ferramentas utilizadas na atualidade para a educação, representando novas maneiras de se atingir seu objetivo principal que é tornar a compreensão e absorção do conhecimento mais fácil e consistente. Com o uso da informática conseguimos uma maior atenção dos aprendizes, formas diferentes de se transmitir a teoria e simular a prática, focalizando todos os sentidos conhecidos para o assunto abordado.

A Multimídia, com todos os seus recursos atraentes, e a Inteligência Artificial, com a qual podemos simular raciocínio e poder de abstração humanos, nos proporcionam hoje a construção de sistemas educacionais cada vez mais envolventes e úteis. Com o propósito não de substituir os professores, mas direcionados para o *auxílio* à aprendizagem, os sistemas educacionais são desenvolvidos na tentativa de conseguir resultados não atingidos com as técnicas pedagógicas tradicionais ou apenas aprofundar estes resultados, como também para permitir a realização de treinamentos e estudos sem a presença ou proximidade obrigatória do mestre humano.

Neste capítulo, serão fundamentados os princípios básicos dos STIs, sistemas voltados à educação com o auxílio do computador e que utilizam técnicas de Inteligência Artificial para flexibilizar o ensino, permitindo uma melhor interação entre o sistema e o usuário (aluno). Será apresentado um breve histórico que abrange desde o surgimento dos sistemas de instrução auxiliada pelo computador até o surgimento dos STIs. Abordaremos algumas arquiteturas desenvolvidas por vários pesquisadores da área, seguindo uma organização cronológica para caracterizar os avanços na área. Analisaremos então, os componentes básicos da estrutura de um STI, bem como discutiremos algumas questões concernentes ao desenvolvimento destes sistemas.

2.2. Histórico

A área de Instrução Assistida por Computador (*Computer-Aided Instruction*, CAI) existe desde a década de 50. A abordagem então utilizada baseava-se na psicologia behaviorista e caracterizava-se pelo conceito de "programa linear", que funciona da seguinte maneira: o sistema apresenta uma determinada unidade de texto, a qual deve levar o aluno um pequeno passo adiante na direção de um comportamento esperado; o aluno então responde de alguma forma, por tentativa e erro ou baseado em conhecimento prévio e o sistema informa se a resposta está correta ou não. Com isso, o aluno pode aprender em seu próprio ritmo, recebendo "feedback" imediato. [VICC 92]

No final dos anos 60 e no início dos anos 70, os pesquisadores interessados no uso do computador na educação sustentavam que um programa tutorial deveria ter uma representação daquilo que se propunha a ensinar, além de um modelo de quem receberia o ensinamento e as estratégias que orientassem o aluno.

Os programas tradicionais de CAI são desenvolvidos através do encapsulamento das decisões pedagógicas em seu código, de forma que a base de conhecimento utilizada é fixa. Este tipo de programa apresenta ainda a limitação de não considerar as características individuais dos alunos.

Vários sistemas foram implementados no final dos anos 70 e durante a década de 80: GUIDON, um tutor construído a partir do sistema especialista MYCIN; SOPHIE I, II e III, para o ensino de diagnóstico de problemas em circuitos elétricos; PROUST, para o ensino de programação Pascal; TUTOR-LISP, para o ensino de programação LISP, entre outros. [VICC 92]

Reconhecendo as limitações dos CAI, os pesquisadores passaram a recorrer a técnicas de representação do conhecimento, métodos de inferência e estratégias de controle, visando buscar soluções para a questão da adaptação dos sistemas tutores aos estudantes, individualmente. Surgiu então a área denominada ICAI (*Intelligent CAI*).

Em 1982, Sleeman e Brown publicaram uma edição especial do *International Journal of Man-Machine Studies* na forma de um livro que batizou a área com o seu nome mais conhecido:

Intelligent Tutoring Systems. Em 1988, outro livro foi publicado, sob a supervisão de Self*: Artificial Intelligence and Human Learning.* A partir deste último, é possível perceber o estado da arte em STI e algumas de suas perspectivas e direções futuras.

Os STI foram definidos por Woolf [WOOL 88] como "sistemas que modelam o ensino, a aprendizagem, a comunicação e o domínio do conhecimento, e devem modelar e raciocinar sobre o domínio do conhecimento do especialista e o entendimento do estudante sobre este domínio".

Os pesquisadores da área de STIs passaram a ter um grande envolvimento com as pesquisas da área de IA, mas a solução dos problemas dos STIs dependem da solução de quase todos os problemas da Inteligência Artificial.

2.3. Definições e Características Principais

Segundo Farjado [FARJ 93], os STIs são mais inteligentes que os programas tradicionais porque possuem um comportamento mais parecido ao de um professor, que flexibiliza a apresentação de seu conteúdo instrucional de acordo com o perfil de cada usuário. O autor considera que, apesar desta característica, os programas não deveriam ser classificados em inteligentes ou não inteligentes, mas sim, possuindo uma maior ou menor capacidade de adaptação ao usuário.

Em Nunes et al. [NUNE 93], os STIs possuem uma clara articulação de conhecimento num domínio limitado e um modelo de aluno que guia o processo de instrução, provêem diagnóstico de erros, possibilitam questionamento por parte dos estudantes e geram a seqüência de instrução de acordo com o perfil de cada aluno.

Os STI são Sistemas Baseados em Conhecimento, cuja maior função é instruir alunos na aquisição de conhecimento em algum domínio [WINK 93].

Para Anderson et al. [ANDE 90], os STIs servem para dois objetivos principais: desenvolvimento de sistemas para automatizar a educação e exploração de questões epistemológicas sobre a natureza do conhecimento que será tutorado e como esse conhecimento pode ser transmitido.

De acordo com Shute [SHUT 94], os STIs são sistemas que comportam-se inteligentemente mas não são inteligentes como as pessoas.

Viccari [VICC 93] coloca que os STIs ou Sistemas Inteligentes de Instrução Assistida por Computador são sistemas em que a IA permite flexibilizar o ensino por computador, possibilitando a participação tanto do aluno quanto do sistema, gerando um ambiente cooperante para o ensino e aprendizagem.

Segundo McCalla [MCCA 94], as técnicas de IA as quais os STIs dependem, ainda não estão maduras para serem usadas de forma satisfatória e os STIs estão tentando copiar a postura do professor em vez de usar a tecnologia computacional de maneira mais natural para a área tecnológica. O autor ressalta ainda que a palavra "tutor" nos STIs indica a flexibilidade no *apoio* ao aprendizado, como faz um tutor, e não no *direcionamento* do aprendizado, como faz um professor.

Os STIs são sistemas nos quais a IA desempenha um papel de relevo, não só por permitir maior flexibilidade no ensino por computador, mas também por possibilitar a participação ativa do aluno e do sistema, gerando um ambiente cooperante para o ensino e a aprendizagem (de ambos os agentes, aluno e sistema). É preciso destacar a importância da interação cooperativa, onde ambos os agentes (tutor/aluno) interagem visando a troca de conhecimento.

A seguir veremos algumas arquiteturas propostas para o desenvolvimento de STIs, bem como as características principais de cada modelo.

2.4. Modelos e Exemplos

Uma arquitetura de software deve separar a funcionalidade dos STIs numa coleção de componentes de software com interface e projetos de execução muito bem definidos, com o intuito de alcançar os seguintes objetivos [WARR 93]:

a) *Isolamento e inserção de tecnologia* – a arquitetura deve permitir a inclusão de novas tecnologias, assim como, diminuir as dependências nas tecnologias existentes;

b) *Facilidade de aquisição* – a arquitetura deve reduzir custos de desenvolvimento e

manutenção associados à construção de múltiplos STIs;

c) *Diminuição da dependência de domínio* – a arquitetura deve diminuir as dependências de domínio dos STIs;

d) *Diminuição das estratégias instrucionais* – a arquitetura deve permitir estratégias instrucionais com impacto mínimo em outros STIs.

Em [COST 98b], vários autores como D. Robston, M. Yazdani, E. Fischetti e K. Warren, propõem uma arquitetura básica representada na figura 2.1, que é constituída por:

- **Modelo do Domínio** – define o conhecimento contido no domínio a ser ensinado;
- **Modelo do Aluno** – é capaz de definir o conhecimento do aluno em cada ponto durante a instrução;
- **Modelo do Tutor** – é responsável pelas estratégias de ensino-aprendizagem adequadas;
- **Interface** – permite a interação do estudante com o STI de maneira eficiente.

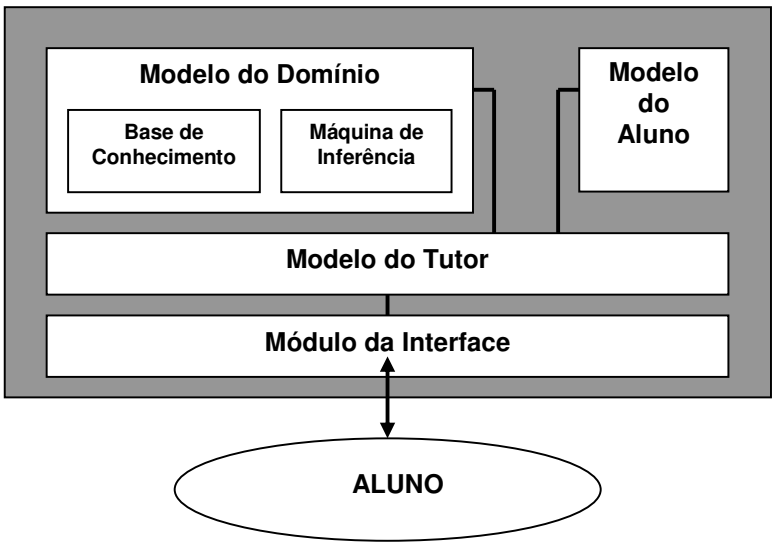


Figura 2.1: Arquitetura Geral dos Sistemas Tutores Inteligentes [FISC 90]

Contudo, analisaremos agora outras arquiteturas de STIs, propostas por vários autores.

Woolf [WOOL 88], em um trabalho sobre a composição dos STI, sugeriu uma arquitetura de quatro modelos: modelo do conhecimento e raciocínio, meio de comunicação, modelo do processo cognitivo e modelo do tutor. A autora afirma que os quatro modelos são considerados como o mínimo para um sistema ser considerado inteligente.

Corredor em 1989, referenciado em [PINT 95], descreve um modelo para construção de STIs composto pelos mesmos componentes básicos, diferindo apenas nas relações entre eles. Veja a figura abaixo:

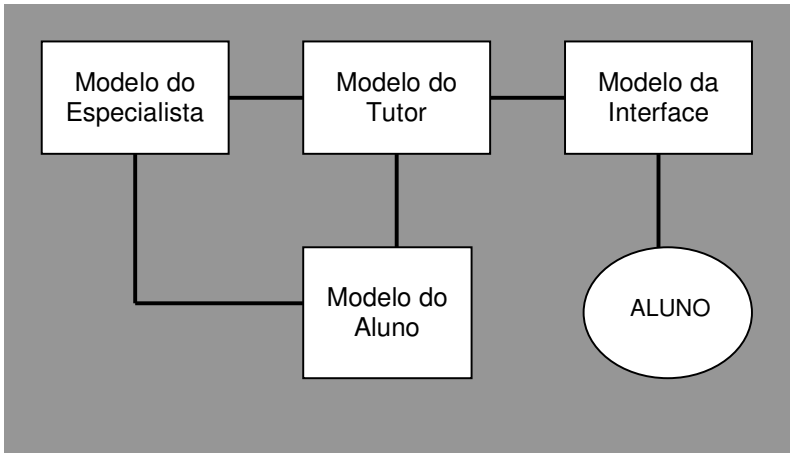


Figura 2.2: Arquitetura de STI proposta por Corredor [PINT 95]

Fischetti e Gisolf [FISC 90], apresentam também um modelo de STI composto por quatro módulos similares aos expostos na arquitetura básica, reforçando as propostas de Woolf.

McCalla e Greer [MARK 93], em 1990, definiram uma arquitetura formada por seis componentes:

a) *Conhecimento do Domínio* – armazena, raciocina e manipula o conhecimento do domínio;

b) *Ensino* – usa o conhecimento de como ensinar;

c) *Comunicação* – apresenta as informações para o estudante e recebe suas respostas;

d) *Conhecimento* do *Aluno* – descreve a compreensão que o estudante possui sobre o

conhecimento e diagnostica suas necessidades;

e) *Aprendizagem* – monitora e ajusta o comportamento do sistema;

f) *Controle* – gerencia as operações do STI.

Em [VICC 92] é proposta uma arquitetura composta pelos quatro modelos mais aceitos na literatura: O módulo do domínio, o módulo do aluno, as estratégias de ensino e o módulo da interface, diferenciando da arquitetura básica pela inserção de mais um componente: o Controle. Tal arquitetura está ilustrada na figura abaixo:

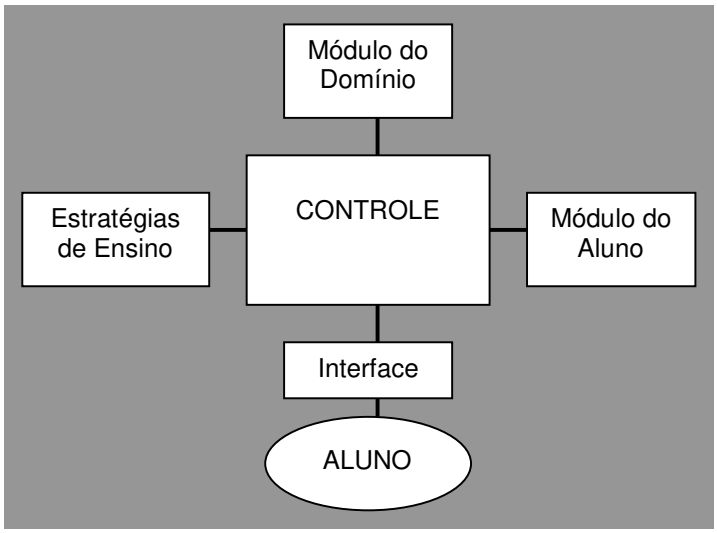


Figura 2.3: Arquitetura de um Sistema Tutor Inteligente [VICC 92]

A explicação para esta separação, segundo a autora, é que "o controle, na prática, está embutido na interface e que as estratégias de ensino, por sua vez, podem estar fixadas e codificadas como parte do mecanismo de controle." No entanto, as estratégias de ensino podem estar codificadas de forma separada do controle. O controle então passa a ter dentre outras funções básicas a de "selecionar uma estratégia de ensino no banco de estratégias". Portanto, neste caso, é útil manter o sistema de controle em um módulo à parte. O controle executa também a seleção do material instrucional, a apresentação deste material ao estudante e o diagnóstico de seu comportamento. Em resumo, o controle gerencia o funcionamento do sistema tutor.

Warren et. al. [WARR 93], apresentam uma arquitetura genérica baseada em tecnologia

orientada a objetos, onde cada um dos componentes da arquitetura é considerado como uma entidade independente e seguem o princípio da POO através da comunicação na forma de mensagens entre os módulos. É enfatizado que tal arquitetura contribui para a redução dos custos de desenvolvimento e manutenção, facilitando a construção de múltiplos STI.

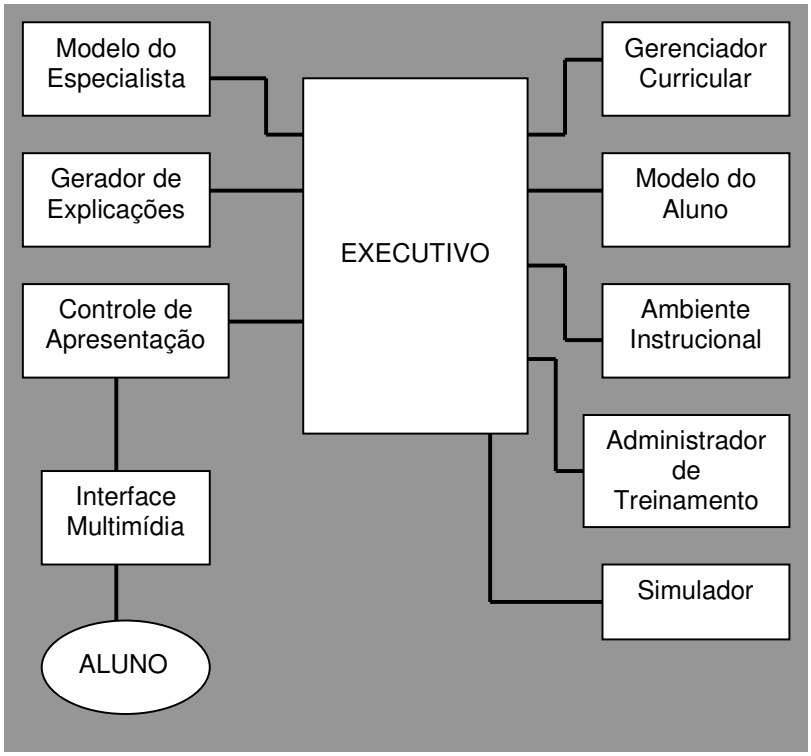


Figura 2.4: Arquitetura genérica de Warren et al. [WARR 93]

O componente chave desta arquitetura é o Executivo, que implementa o controle de execução que guiará o processamento do sistema. Nesta arquitetura, o Modelo do Tutor foi dividido em: gerenciador curricular, controlador de apresentação, gerador de explanação, interface multimídia e ambiente instrucional.

Para testar esta arquitetura, foi implementado um protótipo nomeado MMTS composto pelos quatro componentes principais: executivo, ambiente instrucional, modelo do aluno e modelo do especialista. Neste caso, apesar da modularidade da arquitetura proposta, percebe-se que na prática foram implementados somente os quatro componentes mais comuns da arquitetura geral, ou seja, aqueles que têm servido de base para a grande parte das novas propostas de arquiteturas.

Brusilovsky [BRUS 93] apresenta o conceito geral dos Ambientes Inteligentes de Aprendizado (AIA), ressaltando que tais ambientes seriam um novo tipo de sistema educacional inteligente. Os AIA são compostos pelos quatro elementos básicos acrescidos de um quinto componente, o módulo de ambiente. Este módulo define o tipo de problema apresentado ao estudante e quais as ferramentas disponíveis para resolvê-lo. Contudo, a interface apresentada é orientada a alunos experientes, sendo muito complexa para os novatos. Como solução para tal situação, é apresentada a interface adaptativa, sensível ao nível de conhecimento de cada aluno.

Reinhardt et al. [REIN 95] apresentam a arquitetura proposta por Puppe em 1992 e que também é composta pelos quatro componentes básicos. Sua diferenciação do modelo acima proposto encontra-se na comunicação entre o módulo de interface e o módulo do aluno, como ilustrado na figura abaixo:

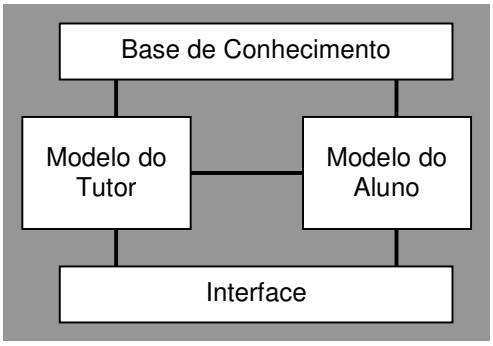


Figura 2.5: Arquitetura proposta por Puppe [in REIN 95]

Notamos que, apesar de pequenas diferenças entre as arquiteturas, seus módulos e suas interligações, destaca-se sempre uma arquitetura básica composta pelos seguintes componentes: o módulo do conhecimento, o módulo do aluno, o módulo do tutor e o módulo da interface. No entanto, esta arquitetura caracteriza a *Abordagem Tradicional* de STIs, sendo que para a *Abordagem de Agentes*, teremos os mesmos componentes, modelados sob uma arquitetura orientada a agentes, onde os mesmos elementos desta arquitetura básica estão presentes de alguma forma, na maioria dos casos.

Detalharemos, na próxima seção, cada um dos componentes que compõem a arquitetura básica de um STI.

2.5. Componentes da Arquitetura Básica

2.5.1. A Base de Conhecimento do Domínio

O modelo de domínio é o componente especialista do tutor constituído pelo material instrucional, por uma sistemática geração de exemplos, pela formulação de diagnósticos e pelos processos de simulação. O material instrucional é organizado, geralmente, numa taxinomia que prevê níveis crescentes de complexidade. As tarefas são organizadas utilizando uma estrutura de árvore e formadas dinamicamente, de acordo com a interação e o trabalho do aluno. [VICC 96]

Este é um componente chave do sistema tutor inteligente. Nele está representado o material instrucional, isto é, o conteúdo que o tutor deverá ministrar. O fato desse conteúdo ser administrado em uma base de conhecimento e não em uma base de dados convencional, é um dos fatores que determinam a diferença entre um STI e um CAI convencional.

É esta base de conhecimento que deve facultar ao sistema a possibilidade de raciocinar sobre a estrutura do conteúdo a ser ministrado, permitindo-lhe então ser mais do que um simples virador de páginas eletrônico. Uma conseqüência evidente desta importância funcional é que a representação de conhecimento utilizada determina fortemente o comportamento do sistema tutor. A forma de representação dependerá, como em todos os sistemas baseados em conhecimento, do domínio de aplicação considerado. O tipo de manipulação a ser efetuada pelo sistema tutor também influencia a forma de representação do conhecimento.

Na construção da base de conhecimento dos sistemas tutores inteligentes são utilizados os mesmos métodos dos sistemas baseados em conhecimento. Encontramos em [COST 98b] e em [FISC 90], uma categorização dos principais métodos de representação do conhecimento:

- ***Lógica*** – uso de expressões da lógica formal para representar o conhecimento através de regras de inferência e prova formal a instâncias do problema;
- ***Regras de Produção*** – são a representação mais efetiva para descrições declarativas do conhecimento. São constituídas pelas regras e pela produção, sendo que a primeira parte da regra é a condição e a segunda é a conclusão, quando a

condição é verdadeira;

- ***Redes Semânticas*** – usam nomes dos objetos e valores em linguagem natural como nós em um gráfico e arcos (ou ligações) indicando as relações entre os vários nós;

- ***Frames*** – espécie de rede semântica onde os nós que são ligados no frame descrevem objetos, eventos e situações estereotipadas;

- ***Meta-regras*** – expressam propriedades de outras regras. São regras que concluem algo sobre outras regras e estão fundamentadas no conceito de meta-conhecimento;

- ***Orientação a Objetos*** – é considerada como uma nova e poderosa ferramenta para a representação do conhecimento. Engloba a teoria dos frames onde cada frame descreve uma classe de objetos. A modularidade, a abstração, a passagem de mensagem, a herança, classes e objetos propiciam um relevante poder à forma como o conhecimento pode ser representado.

Segundo Viccari [VICC 96], o embasamento cognitivo da representação utilizada na base de conhecimento do domínio e no modelo do aluno reflete o caráter multidisciplinar dos STIs enquanto área de pesquisa.

2.5.2. O Modelo do Estudante

Este modelo é responsável por estabelecer um perfil do aluno diagnosticando deficiências, seu nível de conhecimento, formando uma imagem de sua compreensão do conteúdo instrucional, servindo de subsídio para o sistema decidir o quê e como ensinar, de acordo com o nível em que o aluno se encontra. Este módulo representa o conhecimento e as habilidades cognitivas do aluno em um dado momento. É constituído por dados estáticos e dados dinâmicos, que serão de fundamental importância para o tutor poder comprovar hipóteses a respeito do aluno.

Viccari & Oliveira [VICC 92] colocam que, como a avaliação do aluno consiste em uma comparação entre este modelo e a base de conhecimento, é importante que tanto a base

quanto o modelo do aluno sejam apoiados numa teoria cognitiva, que descreva os modelos e processos mentais utilizados pelo estudante para que essa comparação seja viável.

O modelo do aluno pode ser representado apoiando-se em alguns modelos de descrição [COST 98b]:

- ***Modelo Overlay*** – o conhecimento do aluno é representado como um subconjunto da base de conhecimento do sistema tutor. O modelo é construído por comparação do desempenho do estudante com a ação adotada pelo computador baseada no conhecimento do especialista para um mesmo problema. Os erros neste modelo são creditados à ausência de alguma informação presente na base de domínio.

- ***Modelo de Buggy (Modelo de Pertubação)*** – também relaciona o modelo do aluno com a base de conhecimento do domínio. Em [BERT 94], o Modelo de Buggy é conhecido por este nome por relacionar o conhecimento do aluno com uma perturbação do conhecimento especialista. Os erros dos alunos são assumidos pela presença de concepção errôneas de algum conceito. Em geral, este modelo possui uma biblioteca de erros típicos. Viccari & Oliveira [VICC 92] apontam que a vantagem do Modelo de Buggy sobre o Modelo de Overlay reside na possibilidade de representar e reconhecer concepções incorretas por parte do aluno ou ausência de conceitos.

- ***Modelo de Simulação*** - permite prever o comportamento futuro do comportamento modelado, ou seja, prevê a resposta do estudante baseado em seu comportamento.

- ***Modelo de Crenças*** – consiste em um conjunto de crenças refletindo o grau de entendimento que pensamos que o estudante tem sobre um conceito particular.

- ***Modelo de Estereótipo*** – Em geral, classifica o usuário como novato, intermediário ou avançado em uma dada área de conhecimento. O nível de conhecimento do usuário é representado por um conjunto de pares (tópico, valor), com o usuário podendo ter um ou mais estereótipos ao longo da

realização da tarefa.

Segundo Brusilovski (in [PINT 95]), a combinação do Modelo Overlay com o Modelo de estereótipo tem gerado bons resultados. O autor ressalta que o Modelo de Estereótipo se mostra bastante confiável e é mais simples que o Modelo Overlay, porém menos flexível e poderoso.

2.5.3. O Modelo do Tutor

É encarregado de definir e aplicar uma estratégia pedagógica de ensino, ou seja, decidir e guiar o processo de ensino-aprendizagem. Contém os objetivos a serem alcançados e os planos utilizados para alcançá-los. Em [ASAN 91], este modelo é responsável pelas tarefas de selecionar problemas, monitorar e criticar o desempenho, prover assistência quando requerido, selecionar material instrucional a ser apresentado ao estudante. Integra conhecimento acerca do método de ensino e do domínio a ser ensinado. Também chamado de Modelo Instrucional e de Módulo Tutorial.

Segundo Zorita, em [VICC 92], pode ser identificado um conjunto de atividades estratégicas envolvidas no processo de tutoria, que foram diferenciadas como operativas (no sentido de guiar o aluno) e didáticas (alcançar objetivos). São elas:

- **Operativas**:
  i. *Contextualizar o aluno* – manter o aluno informado sobre o ciclo de atividades que está sendo seguido;
  ii. *Motivar e manter a atenção do aluno* – lançar estímulos e propostas que ajudem a manter o interesse do aluno durante todo o processo de tutoração;
  iii. *Guiar a atuação do aluno* – conduzir a resolução dos exercícios, fornecendo dicas e conselhos de modo a prevenir erros constantes que poderiam desestimular o aluno.
- **Didáticas:**
  i. *Apresentar o conhecimento do domínio* – proporcionar uma formação tanto a nível teórico como prático do domínio de aplicação do sistema,

explicando conceitos, mostrando operações e propondo exercícios práticos;

ii. *Avaliar o conhecimento adquirido* – comprovar o conhecimento adquirido através de testes e exercícios práticos;

iii. *Tratamento de erros* – complementar o conhecimento considerado deficiente.

A organização do modelo de tutoração depende de como é conduzido o processo de interação no STI.

Um ciclo básico de tutoração pode ser formado pelas seguintes tarefas [CAVA 91]:

1. Aceitar a resposta do aluno;
2. Comparar a resposta do aluno com a resposta proposta pelo modelo do conhecimento;
3. Modelar o aluno;
4. Diagnosticar o nível mental do aluno;
5. Fornecer esta informação ao modelo de tutoração;
6. Executar a ordem proveniente do modelo de tutoração;
7. Apresentar as possibilidades para a próxima interação.

Segundo Viccari & Moussale [VICC 90], os STI apresentam, em geral, o modelo tutorial socrático, que prevê situações de diálogo a partir de um fato conhecido do aluno, levando-o a aperfeiçoar este conceito através de explorações de contradições e da formulação de inferências corretas a partir do conhecimento inicial. Entretanto, Glanzmann [GLAN 95] apresenta outras estratégias educacionais para a especificação da apresentação do material a ser ensinado, a saber:

- *Método de Treinamento*: propõe um ambiente em que os alunos se divirtam e tenham prazer em aprender. O ambiente explora a solução de problemas gerais através de atividades como jogos;
- *Método Orientador*: o sistema atua por solicitação do estudante quando este requisita palpites, expansões ou críticas.

2.5.4. O Modelo da Interface

Este modelo tem por objetivo facilitar a operacionalidade do sistema, bem como, torná-lo atrativo e motivador, intermediando a comunicação entre o computador e o aluno, apresentando material instrucional, fazendo o mapeamento entre a representação interna do sistema contida nos módulos e traduzindo-a numa linguagem de interface compreendida pelo aluno. Segundo Magalhães [MAGA 95] esta é uma das tarefas mais complexas ao se implementar um tutor inteligente, pois a interface deve estimular a aprendizagem, buscando estabelecer uma interação simples e amigável. Para isso, usam-se recursos como simulação, animação, menus, ícones, janelas, processamento de linguagem natural ou pseudo-natural, capacidade de representação gráfica, entre outros.

Uma boa interface é vital para o sucesso de qualquer sistema interativo e Sistemas Tutores Inteligentes não constituem exceção. Pelo contrário, pode-se dizer que a questão da interação cresce de importância nesta classe de sistemas, pois é na interação que o sistema tutor exerce duas de suas principais funções, a apresentação do material instrucional e a monitoração do progresso do estudante através da recepção da resposta do aluno. Dessas duas funções, podemos derivar alguns objetivos a serem cumpridos pelo módulo de interface:

1. **É necessário evitar que o estudante se entedie** – ou seja, é preciso riqueza de recursos na apresentação do material instrucional;
2. **É desejável que haja facilidade para a troca na iniciativa do diálogo** – o estudante deve poder intervir facilmente no discurso do tutor e vice-versa;
3. **O tempo de resposta deve, evidentemente, permanecer dentro de limites aceitáveis**;
4. **A monitoração deve ser realizada o máximo possível em *background*** - para não onerar o estudante com questionários excessivos mas também respeitando a barreira do tempo de resposta.

As possibilidades na apresentação do material instrucional têm recebido um significativo avanço a partir da utilização de sistemas de hipertexto e hipermídia. Em seu nível mais básico, um sistema de hipertexto é um sistema que permite conectar telas de informação através de ligações entre uma palavra ou frase e um texto com informações relativas àquela

palavra ou frase, ou seja, partir de um fragmento de texto clicável com o mouse é possível visualizar um texto com mais informações sobre o primeiro. O termo hipermídia é utilizado quando o material envolvido compreende também recursos multimídia como gráficos, som, animação, etc. A WWW (The World Wide Web) é hoje um exemplo comum de hipermídia onde os recursos estão espalhados por toda uma rede mundial de computadores, interligados por *links*, vínculos virtuais entre os arquivos. Essa variedade de recursos, aliada à possibilidade de percorrer o material de maneira vinculada à semântica do conteúdo, fazem dos sistemas de hipermídia uma ferramenta de alto potencial para apresentação do material instrucional em Sistemas Tutores Inteligentes.

Segundo Viccari [VICC 96], a monitoração do progresso do estudante pode ocorrer em dois níveis:

a) No nível da análise do histórico do estudante, ou seja, de uma sessão para outra;

b) No nível do diagnóstico circunscrito de uma sessão.

Evidentemente, o funcionamento da interface está mais relacionado ao segundo nível. E esses níveis devem trocar informações entre si. Haverá momentos em que o diagnóstico local deverá levar em conta o histórico armazenado no modelo do estudante, mas só se o modelo do estudante tiver histórico (o que nem sempre é verdadeiro). Além disso, uma busca no histórico pode ser demorada, comprometendo o tempo de resposta na interação.

Uma barreira tradicional e que é comum com a IA é o processamento de linguagem natural. Uma vez solucionados determinados problemas neste campo, a comunicação entre o homem e o computador certamente será bem mais fácil e eficiente. Mas apesar das limitações atuais, alguns sistemas conseguem trabalhar com expressões verbais dentro de uma linguagem pseudo-natural [FISC 90].

2.6. Arquiteturas Multi-Agentes para STIs

Nesta seção comentaremos alguns exemplos de STIs desenvolvidos utilizando arquiteturas multi-agentes. Estes exemplos foram escolhidos por terem algo em comum com o nosso trabalho. O primeiro, por se tratar de um ambiente de aprendizagem musical, o segundo,

um ambiente de aprendizagem implementado através da linguagem JAVA com opção para educação via rede. O MATHEMA, último trabalho desta seção, é comentado resumidamente pois dedicou-se um capítulo ao detalhamento e identificação das partes do mesmo, para serem utilizadas como modelo para o MHITS.

2.6.1. O Expert Piano

O Expert Piano, [GLAN 95] é um ambiente educacional apoiado em tecnologia computacional e musical para auxiliar o estudo de piano e música, com direcionamento especialmente voltado para a prática musical.

O ambiente Expert Piano tenta simular o comportamento de um professor que esteja assistindo à interpretação do aluno, analisando o seu desempenho e mostrando quando este incidir em erros de execução musical. Segundo Glanzmann, o uso de técnicas de IA em um sistema permite torná-lo mais flexível e adaptado às necessidades de cada aluno.

2.6.1.1. Objetivos do Expert Piano

Os propósitos principais deste ambiente são a análise do desempenho do aluno, detectando, quando for o caso, erros de execução musical e a apresentação de apontamentos e sugestões sobre como o estudante deve proceder para a correção dos mesmos.

O domínio específico é a área de educação musical, trazendo para o ambiente computacional tarefas comuns a esta área, executadas às vezes pelo professor, às vezes pelo aluno, às vezes pelos dois. Alguns exemplos são: ensinar instrumento (neste caso, o piano), programar estudos, escolher peças musicais, estudar instrumento, estudar técnica, estudar obra musical, acompanhar aproveitamento do aluno, dentre outras.

2.6.1.2. Modelagem e Especificação do Expert Piano

Neste sistema, o conhecimento é modelado em três camadas: domínio, inferência e tarefa. Na Camada de Domínio é representado o conhecimento estático do domínio do problema, com os conceitos e relações independentes do raciocínio utilizado. A Camada de

Inferência possui o conhecimento necessário para inferir novos fatos a partir do conhecimento do domínio do problema. A Camada de tarefas representa a descrição de quando realizar as inferências descritas na camada de inferência.

Na Especificação de Projeto temos o Esquema de Representação e Inferência do Conhecimento, onde o conhecimento é representado em forma de regras de produção. O Modelo Lógico, baseado no Kads-estendido, é construído a partir a partir do Modelo de Especialidade, sendo composto pelo Diagrama Heurístico do Raciocínio e pelo Diagrama do Domínio do Problema.

A modelagem física é composta pelo Diagrama de Interface com o Usuário, construído a partir da hierarquia dos comandos e a interação do sistema com o usuário, o Módulo Ajuda, para auxílio às dúvidas de operação e manuseio do sistema, o Modo Professor, acessado apenas pelo professor para manutenção das bases de dados do ambiente, o Módulo HiperMusic, organizado em estrutura de hipertexto contendo aspectos teóricos básicos sobre teoria musical e sobre as peças musicais estudadas. Este último não foi implementado na primeira versão do sistema. Além destes, temos o Módulo Partitura, para apresentação dos dados musicais em formato de notação musical convencional, o Módulo Sessão de Estudos e o Módulo Ouvir Músicas.

Além do Diagrama de Interface, a especificação explicita ainda o Diagrama de Explicação do Raciocínio e o Modelo de Implementação do Sistema.

2.6.1.3. Protótipo do Expert Piano

O protótipo deste ambiente, apresentado em [GLAN 95], foi implementado com a utilização da ferramenta de autoria *Asymetrix Multimedia ToolBook 3.0*, um ambiente normalmente usado para o desenvolvimento de aplicativos multimídia para o *Windows*, integrada com rotinas escritas em *Borland C++*.

A máquina de inferência foi implementada em C++, adaptada de exemplos de livros. As rotinas em C++ são chamadas via DLL pelo módulo principal desenvolvido em *TollBook*. A memória de trabalho foi representada em arquivos padrão *dBase* (.dbf). A interface com o usuário segue os padrões do Windows, contendo ícones, botões, janelas e menus, que

simplificam as escolhas de opções, a disponibilização de recursos e facilitam o diálogo com o usuário. O ambiente dispõe ainda de um controle de segurança para acesso ao sistema e a determinados módulos, não permitindo a utilização por pessoas não autorizadas.

2.6.2. O SEI – Sistema de Ensino Inteligente

Para suprir uma demanda em cursos de Introdução a Computação foi desenvolvido por Tedesco, Barros e Souza [TEDE 97] o SEI – Sistema de Ensino Inteligente – um Sistema Tutor Inteligente modular, extensível e portátil. Baseado em uma sociedade de cinco agentes (Estudante, Domínio, Tutor, Comunicador e Controlador), o SEI foi implementado na linguagem JAVA para garantir a portabilidade entre plataformas e execução através da World Wide Web.

Visando resolver problemas como adaptar os componentes do sistema a um domínio de aplicação diferente do original além de não dirigir o desenvolvimento do sistema apenas para uma plataforma específica, os STIs devem, segundo [TEDE 97], buscar modularidade e portabilidade como características principais em sua arquitetura, a fim de tornarem-se viáveis comercialmente.

2.6.2.1. Objetivos do SEI

Após entrevistas com alunos de graduação de alguns cursos, visando uma avaliação da sua motivação em relação a um curso de Introdução a Computação, os objetivos do sistema foram definidos como:

1. Promover o aprendizado dos conceitos básicos de IC;
2. Adaptar-se às características particulares do estudante corrente;
3. Monitorar o desempenho do estudante fornecendo caminhos alternativos de aprendizado;
4. Sugerir seqüências de currículos para o estudante seguir;
5. Ser um sistema extensível e portátil, fácil de manter e refinar.

2.6.2.2. Arquitetura do SEI

O sistema SEI foi desenvolvido com duas bases de conhecimento: os Modelos do Domínio e do Estudante além de cinco agentes, Estudante, Domínio, Tutor, Controlador e Comunicador.

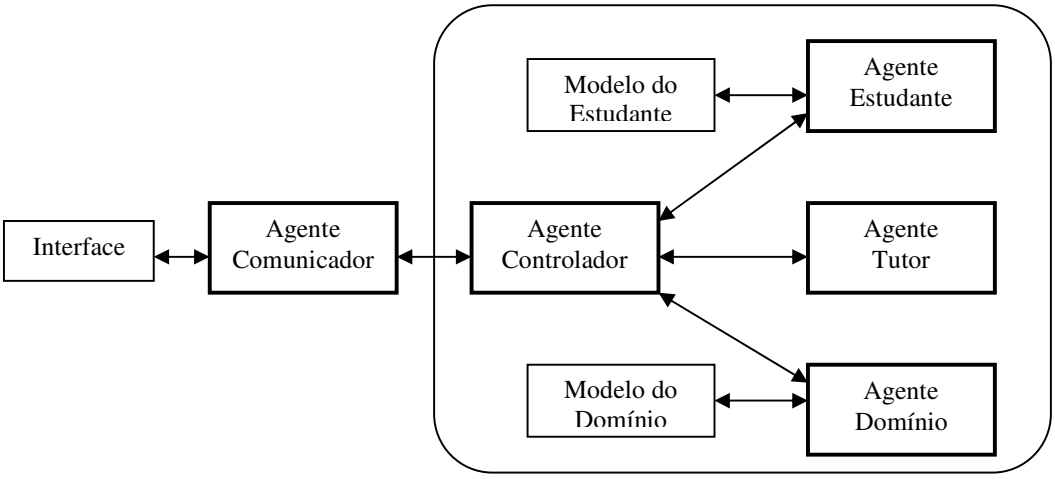


Fig.ura 2.6 Arquitetura do Sistema SEI

O **Modelo do Estudante** armazena informações relativas ao estudante corrente e dispõe de três estereótipos para acomodar as diferenças relativas ao nível de conhecimento prévio do aprendiz: (*1*) Usuário Especialista, (*2*) Usuário Casual e (*3*) Usuário Leigo. Esta atribuição é feita na primeira interação com o sistema. A atualização deste modelo é feita a cada passo da interação do aluno com o sistema.

O **Modelo do Domínio** armazena o conteúdo do curso. Está organizado como uma rede de frames que possui quatro níveis de abstração: Curso, Lições, Tópicos e Apresentações. O sistema apresenta as informações em três níveis de aprofundamento, de acordo com as categorias de usuários citadas anteriormente.

O **Agente Controlador** controla o funcionamento do sistema como um todo gerenciando a troca de informações entre os agentes. É modelado através de regras de produção. Este agente mantém um registro de todos os agentes do sistema, serviços disponíveis e localizações, além de controlar a comunicação entre eles.

O **Agente Estudante** manipula informações relacionadas aos estudantes e determina o perfil do estudante corrente. Para cada sessão tutorial, um modelo de estudante do aluno é criado, caso seja a primeira interação, ou recuperado para que o sistema execute as suas tarefas de acordo com as interações anteriores.

O **Agente Domínio** recupera informações do Modelo do Domínio e avalia as respostas do estudante aos exercícios propostos, auxiliando a determinação do desempenho e do perfil do estudante corrente.

O **Agente Tutor** toma as decisões pedagógicas, ou seja, determina *o que* o sistema vai ensinar, *como* e *quando*. Também determina as estratégias de resposta ao aluno de acordo com o seu desempenho e perfil, raciocinando com base nos Modelos do Estudante e do Domínio. O conhecimento deste agente é modelado através de regras de produção, agrupadas de acordo com vários aspectos da interação como motivação do estudante, desempenho, estratégias pedagógicas, etc.

O **Agente Comunicador** trata os eventos da interface do sistema apresentando ao aluno o conteúdo, informações complementares, exercícios a serem respondidos e comentários do Tutor. Além disso captura as respostas do aluno ao sistema. Está implementado em um applet JAVA, para aumentar o alcance do STI com a sua disponibilização através da WWW.

É através da **Interface** que acontece toda a comunicação com o sistema. Possui as seguintes funcionalidades: (1) *Janela de Apresentação*, para o conteúdo do curso, (2) *Janela de Navegação*, para acesso à estrutura curricular do curso, (3) *Janela de Informações Complementares*, para informações adicionais, (4) *Janela de Comentários*, para respostas do Tutor às ações do aluno, (5) *Ajuda*, para auxílio sobre a utilização do sistema e (6) *Menu "Perguntar"*, para o aluno solicitar informações mais detalhadas durante o curso.

O SEI comunica-se ainda através de relatórios sobre o progresso dos alunos, seus erros e desempenho, e através dos usuários categorizados em dois tipos, *Estudantes*, que podem acessar o sistema individualmente ou associados a uma turma ou *Professores,* controlando o processo de ensino através da supervisão do comportamento da turma frente ao SEI.

2.6.2.3. Protótipo do SEI

O SEI está sendo implementado em JAVA, sendo que cada um dos agentes representa um thread que são executados de maneira contínua e se comunicam através de trocas de mensagens KQML [TEDE 97]. O seu funcionamento está resumido em:

1- O usuário acessa o agente Comunicador através da interface do sistema utilizando qualquer browser WWW habilitado para JAVA. O Comunicador é executado na máquina do cliente localizada em qualquer ponto da Internet e se comunica com outros componentes do sistema através de *scripts Perl* e de mecanismos *CGI* (Common Gateway Interface);

2- Quando as mensagens provenientes do Comunicador chegam ao sistema (no servidor), o Controlador as direciona para os agentes capazes de prover os serviços solicitados. A comunicação entre os agentes dentro do servidor se dá através do protocolo RMI (*Remote Method Invocation*) que é parte de JAVA.

A construção do SEI como uma sociedade de agentes inteligentes cooperantes dá margem a outras extensões para o trabalho com STIs. Um STI desenvolvido como o sistema SEI (sobre a WWW e utilizando uma plataforma de software amplamente disponível, de baixo custo e arquitetura neutra) aumenta enormemente o potencial destes sistemas para o ensino e treinamento à distância.

O sistema necessita ainda de testes exaustivos, avaliações e ajustes, com a intenção de melhorar a sua eficiência e seu potencial de reutilização, através do refinamento da estrutura da sociedade de agentes e de seus mecanismos de raciocínio.

2.6.3. O MATHEMA

O MATHEMA é um ambiente interativo de aprendizagem, sendo concebido para possibilitar um ensino adaptativo e seus desdobramentos no processo de aprendizagem.

Este ambiente é de fundamental importância pois serviu como modelo para especificação de alguns aspectos do nosso sistema. Portanto, não entraremos em mais detalhes nesta seção. O próximo capítulo abordará o MATHEMA e os principais aspectos deste ambiente, suas características e detalhes de modelagem e especificação.

Antes abordaremos, nas próximas seções, importantes discussões sobre os aspectos relevantes ao desenvolvimento de STIs, a problemática e falta de métodos para o seu desenvolvimento e avaliação de qualidade, além das conclusões sobre este capítulo.

2.7. Aspectos Relevantes ao Desenvolvimento de STIs

Em [CORR 93] encontramos recomendações e indicações que podem auxiliar no processo de desenvolvimento de STIs:

1) É necessário que o grupo de desenvolvimento do sistema seja multidisciplinar, isto é, seja formado por profissionais não só da área de informática como também de outras áreas como pedagogos, psicólogos, especialistas no domínio do conhecimento escolhido, etc. A troca de conhecimento entre estes profissionais é proveitosa, tanto a nível pessoal como a nível de projeto;
2) As bases de conhecimento criadas devem conter uma estrutura que suporte futuras ampliações;
3) O envolvimento de professores especialistas na área gera importantes contribuições a partir de suas experiências e reflexões, auxiliando na construção de estratégias que facilitem a aprendizagem na área escolhida;
4) Certas etapas do desenvolvimento devem ser classificadas por serem seqüenciais. São elas:
    - Seleção do tema;
    - Determinação dos objetivos instrucionais;
    - Definição da estrutura do conhecimento: níveis e tipos, sua relação, sua forma de representação e seu uso;
    - Determinação das estratégias de tutoria que se consideram para facilitar a aprendizagem do aluno;
    - Definição do modelo que se usará para determinar a experiência no domínio, as dificuldades e os erros dos estudantes;
    - Caracterização da forma de comunicação entre o estudante e o sistema;
    - Seleção de hardware e software adequados ao desenvolvimento e,

- Uso e avaliação do produto de software produzido.

5) É necessário implementar um modelo do estudante em que o tutor modele a solução de problemas do aluno, identificando os seus erros de conceituação e suas possíveis causas, para poder ajudá-lo a superar as suas dificuldades.

No entanto, o que a experiência tem nos mostrado é que as pesquisas na área de tutores inteligentes se limitam aos protótipos desenvolvidos e que alguns pesquisadores não se sentem motivados a concluir seus trabalhos. Discutiremos na próxima seção, os aspectos que dificultam o desenvolvimento de STIs em geral, incluindo as dificuldades particularmente relevantes ao desenvolvimento de STIs no domínio da Música.

2.8. Problemática no Desenvolvimento de STIs

Nos últimos cinco anos, os STI tem sido objeto de intensa pesquisa, o que pode ser medido através do crescente número de artigos publicados em congressos e revistas, tais como: AI-ED, ED-MEDIA, *Journal of Artificial Intelligence in Education, International Conference of Intelligent Tutoring Systems*. Mesmo assim, a utilização dos STI em situações educacionais reais permanece restrita.

Segundo Costa [COST 98a], os principais fatores envolvidos na baixa disseminação de produtos de software inteligentes são o seu alto custo e o longo tempo requerido para o seu desenvolvimento. Além disso, a necessidade de envolvimento de equipes multidisciplinares e a falta de métodos de desenvolvimento e de métodos de avaliação de qualidade são considerados como pontos críticos. Em se tratando de STIs no domínio musical, incluem-se aí dificuldades na representação do conhecimento em música além de aspectos concernentes ao modelo pedagógico que melhor se adaptem ao ensino de música, mais precisamente de Harmonia Musical.

Ressaltaremos agora alguns dos fatores que dificultam o desenvolvimento e o uso dos STIs atualmente.

2.8.1. A Falta de Métodos para o Desenvolvimento de STIs

A utilização de métodos tradicionais de desenvolvimento de software não parece atender à especificidade do desenvolvimento de sistemas inteligentes. Enquanto os sistemas tradicionais lidam com dados, os sistemas com algum tipo de inteligência manipulam conhecimento, principalmente, heurísticas.

O modelo do domínio, componente da arquitetura básica dos STI, pode ser considerado como um Sistema Baseado em Conhecimento, o que nos conduz aos métodos de desenvolvimento de SBCs, que poderiam ser adaptados e estendidos para apoiar o desenvolvimento de STIs. Dentre algumas experiências já realizadas neste sentido, podemos citar o Expert Piano, visto no tópico anterior, que utilizou o KADS-estendido (Knowledge Analysis and Design Structure) [WERN 95] para especificação e modelagem de um ambiente educacional que auxilia o estudo de piano e música. Entretanto, as experiências ressaltaram a falta de apoio oferecida tanto pelo KADS quanto pelo KADS-estendido para a modelagem dos módulos do aluno e do tutor.

Observa-se que os STI vêm sendo desenvolvidos de maneira empírica, sem o apoio de metodologias específicas, sem critérios de avaliação precisos ou referências a documentação do sistema. Observa-se que, em geral, as referências sobre métodos de desenvolvimento de STI possuem uma abordagem muito mais próxima da IA e Engenharia do Conhecimento do que propriamente da Engenharia de Software [ELOR 96] [ALEX 96].

Segundo Costa [COST 98a], a inexistência, até hoje, de um método de desenvolvimento de STI apoiado nos conceitos da Engenharia de Software, talvez se deva à própria estrutura dos STI, já que muitas das dificuldades que se apresentam na sua construção relacionam-se à pretensão de se implementar uma modelagem completa, tanto do modelo do domínio, como do modelo do aluno. No caso do modelo do domínio, as bases de conhecimento são, em geral, muito específicas e de difícil reutilização em outros domínios. Já no modelo do aluno, a determinação do seu nível de conhecimento, a cada momento do processo de tutoria, apresenta-se como um dos grandes desafios da área.

Farjado [FARJ 93] recomenda que a incorporação de inteligência nos STI seja feita de forma gradual, de tal maneira que se possa ver o funcionamento do sistema antes que se tenha todos os requisitos para o ensino de algum tema. Outros autores, como Eliot & Woolf [ELIO 96], recomendam o desenvolvimento iterativo, onde há uma estreita dependência entre

o planejamento do STI e sua codificação em alguma linguagem de programação.

É indiscutível a existência de uma lacuna entre a fase de concepção do STI até a existência de um protótipo operacional. Mostra-se fundamental buscar metodologias que tornem seu processo de desenvolvimento menos custoso e passível de ser avaliado a partir de critérios bem definidos, gerando produtos de alta qualidade e confiabilidade.

Nota-se a necessidade de se intensificarem as pesquisas na área quando se observa que os métodos de desenvolvimento de SBCs não possuem potencial para serem aplicados diretamente nos STIs. Modelos de ciclo de vida, métodos de desenvolvimento, técnicas para estimativa de custos, controle de qualidade, documentação e integração de equipes multidisciplinares, poderiam ser trazidos a partir da adaptação de instrumentos da Engenharia de Software para a área de STIs, normalizando a criação destes sistemas, diminuindo os custos e a complexidade de tarefas.

2.8.2. A Falta de Métodos de Avaliação da Qualidade para STIs

São encontradas na literatura algumas abordagens para a avaliação de STI e percebe-se que, em geral, são bastante dependentes do processo de desenvolvimento adotado. Na falta de métodos específicos de avaliação, experiências isoladas tentam verificar alguns aspectos que influenciam o desenvolvimento de STI, gerando indicativos preliminares.

Em [MARK 93] temos a associação de técnicas de avaliação de software, tais com inspeção e testes, aos STI, mas não é encontrada uma descrição de como estas avaliações deveriam ser aplicadas. Em [GEIS 92], a verificação formal e a validação são essenciais para que os sistemas com algum tipo de inteligência sejam aceitos em áreas críticas, e nesse sentido, descrevem uma metodologia de desenvolvimento interativa, similar ao modelo espiral, utilizando prototipagem rápida que permite a verificação e validação dos sistemas. Neste caso, os autores apresentam os nomes das etapas sem apresentar nenhuma descrição prática. Em [REGI 94] é enfatizado que o STI deve ser avaliado durante e ao final de seu desenvolvimento, verificando-se se os objetivos técnicos e pedagógicos foram atingidos, assim como seu desempenho em ambientes reais de aprendizagem. Mais uma vez, não há uma descrição clara das idéias ou citações de experiências reais.

Percebe-se que a imaturidade das questões relativas a qualidade de STIs, a complexidade de suas arquiteturas aliadas à existência de poucas experiências desenvolvidas, não favorecem o destaque de nenhuma metodologia específica de avaliação de processos nem de produtos.

2.8.3. A Falta de Modelos Pedagógicos para STIs

No desenvolvimento de um software educacional notamos a necessidade de, além dos aspectos técnicos, uma adequação pedagógica ao contexto utilizado como domínio do conhecimento.

Em [CORR 93], através de experiências em desenvolvimento de STIs, observa-se que as possibilidades pedagógicas e didáticas de produtos de software com as características dos STIs são extensas. Nestas experiências, o autor identificou a importância das características individuais de cada aluno e também reconheceu que a instrução deve ser individualizada, de forma a facilitar ao aluno a criação de estruturas conceituais e metodológicas adequadas a sua capacidade e interesse. Percebe-se então a necessidade de uma profunda reflexão sobre as formas de ensino e estilos de aprendizagem, que são operacionalizados em conjunção pelo módulo do aluno e módulo do tutor.

Segundo Costa em [COST 98a], observa-se na literatura relacionada a esta área do conhecimento, que existem poucas referências à utilização de teorias pedagógicas na concepção dos STIs, apesar da importância do tema. No entanto, a valorização da aplicação de teorias pedagógicas nos STIs não é uma unanimidade. Em [REGI 94] vemos que prescrições ou teorias instrucionais não apresentam mecanismos de abordagem da aquisição e representação do conhecimento e mesmo quando o projeto de um STI é cuidadosamente ligado a uma teoria pedagógica consistente, seu valor só poderá ser determinado a partir de testes empíricos.

Estas abordagens convergem para questões ligadas, principalmente, à modelagem das estratégias pedagógicas utilizadas para os diferentes tipos de alunos, ou seja, como interagir com o aluno de forma individual e mantendo seu interesse constante, apresentando os tópicos escolares de modo atrativo, criativo e integrado a seu nível de conhecimento. Em [BALA 94] é

enfatizado que diferentes direções para o progresso da área de modelagem do aluno e da tutoria estão abertas, e vão da busca de modelos psicologicamente válidos, até propostas onde os modelos de aluno não são considerados como componentes importantes para um bom desempenho do produto.

2.9. Conclusões

Os Sistemas Tutores Inteligentes possibilitam um ensino auxiliado por computador flexível e inteligente na forma de apresentação do conteúdo, na avaliação de qual material será exposto ao aluno dependendo do seu nível de aprendizado, na sua forma de comunicação onde predominam o entendimento do aluno e a satisfação deste no uso de um sistema agradável e atraente.

É percebido, ainda que lento, o progresso da pesquisa nesta área, devendo ser incentivada, como também a expansão no uso dos sistemas tutores inteligentes como sistemas auxiliares na educação, utilizados sozinhos ou paralelamente à aplicação de aulas normais por professores humanos.

Muitos problemas ainda são encontrados em relação ao desenvolvimento, validação da qualidade e utilização de modelos pedagógicos tradicionais nos STIs, mesmo estando em andamento pesquisas relacionadas a estas áreas. Estes problemas estão relacionados ao desenvolvimento de STIs em todos os domínios do conhecimento, incluindo o domínio da música. Ao se trabalhar com este domínio, deve-se incluir dificuldades relativas a representação e manipulação do conhecimento musical em todos os seus aspectos.

No próximo capítulo abordaremos o MATHEMA, ambiente utilizado como modelo para aspectos da representação do conhecimento do domínio e do desenvolvimento do nosso sistema.

Capítulo 3
O MATHEMA e a Representação do Conhecimento em Harmonia Musical

3.1. Introdução

Segundo [COSTA 97b], as interações tutoriais entre duas entidades principais, uma máquina, no papel de tutor, e um humano, como um aprendiz, são foco principal em boa parte das pesquisas na concepção de sistemas educacionais inteligentes, STIs. O objetivo maior é fazer com que o sistema tutor facilite, no processo de interação, a aquisição de conhecimento por parte do aprendiz, durante as atividades pedagógicas. Para isso, o sistema tutor deve contar com bases de conhecimento e com mecanismos de raciocínio, sobre as ações do aprendiz e sobre essas bases, para adaptar-se, segundo a sua percepção, às necessidades individuais do aprendiz, no processo de interação dinâmica. A IA aparece, neste momento, provendo técnicas para representação e manipulação do conhecimento envolvido.

Em sistemas baseados no domínio da música, bem como em outros domínios do conhecimento humano, o maior desafio está em como lidar com a complexidade deste conhecimento que o sistema deve se utilizar para promover uma interação apropriada com o aprendiz humano.

Descreve-se sucintamente, neste capítulo, o ambiente interativo de apoio ao ensino e aprendizagem denominado MATHEMA [COSTA 97b] que, neste trabalho, estará utilizando a Harmonia como domínio de aplicação. Tomando como base resultados alcançados em [TEIX 97] e seguindo o MATHEMA como alternativa para a modelagem do sistema e principalmente para a especificação do domínio do conhecimento, pretende-se desenvolver a primeira versão de um STI o ensino de Harmonia Musical. O MATHEMA foi escolhido como base para este trabalho justamente por prover um modelo adequado à representação do domínio do conhecimento abordado e à implementação dos módulos do sistema, discutidos posteriormente.

3.2. O Ambiente MATHEMA

MATHEMA é um ambiente de aprendizagem interativo, sendo concebido para possibilitar um ensino adaptativo e seus desdobramentos no processo de aprendizagem. O ensino adaptativo é aqui visto como conseqüência do processo de interações cooperativas envolvendo os seus componentes (*Aprendiz, Tutores*) em atividades baseadas em resolução de problemas. Nesta perspectiva, o MATHEMA propõe-se a prover princípios e uma arquitetura alternativa, necessários para orientar o desenvolvimento de STI particulares [COSTA 97b].

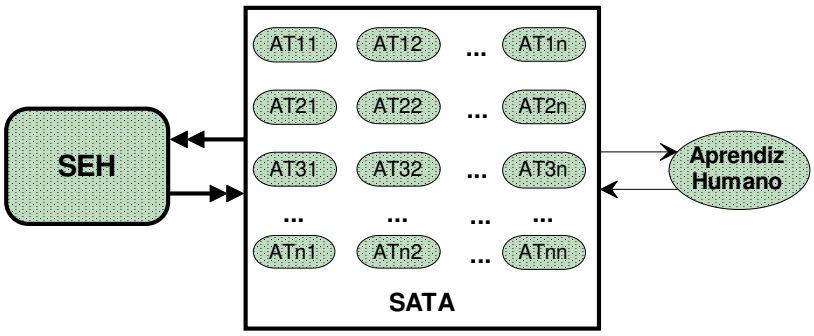


Figura 3.1: arquitetura simplificada do MATHEMA

A arquitetura simplificada do MATHEMA é mostrada na Figura 3.1. Ela consiste essencialmente em três componentes:

1. *Aprendiz Humano*: agente interessado em aprender algo sobre um dado domínio, trabalhado no ambiente MATHEMA.

2. *Sociedade de Agentes Tutores Artificiais* (SATA): conjunto de agentes que algumas vezes cooperam entre si a fim de promover a aprendizagem de um dado aprendiz em atividade de ensino. SATA é definida, conforme mencionado anteriormente, em função de uma visão sobre o conhecimento do domínio, resultando em uma série de subdomínios, obedecendo a um certo critério. Essa idéia foi inicialmente inspirada nas reflexões contidas em "Sociedade da Mente", de Marvin Minsky [MINS 85]. Segundo Minsky, a inteligência emerge da combinação de agentes mentais, cada um responsável por um pequeno processo.

3. *Sociedade de Especialistas Humanos* (SEH): fonte de conhecimento externa ao sistema computacional (algo como um agente oracular) para a SATA. Dessa Sociedade é

requerida a manutenção da SATA e eventual assistência aos aprendizes.

O objetivo principal, de nível mais alto, em um sistema educacional como o MATHEMA é capacitar o Aprendiz a adquirir conhecimento como resultado de sua interação com o tutor. Compreende-se por esta interação a troca de mensagens entre o sistema e o aprendiz. É fundamental o investimento na qualidade relacionada à natureza e conteúdo dessas mensagens, bem como em relação aos seus processamentos para, por exemplo, escolher um conteúdo adequado no momento das intervenções. Para tanto, precisa-se definir um ambiente suficientemente rico para realizar tal pretensão. Portanto, torna-se imprescindível a utilização de interfaces multimídia, principalmente em se tratando de ambientes voltados ao ensino de aspectos relacionados à Música.

Algumas exigências de qualidade desejáveis em direção à concepção de um bom ambiente para ensino/aprendizagem tornam-se necessárias. Para um melhoramento e efetividade no processo de interação entre o Sistema e o Aprendiz, deve-se incluir: *(i) o conhecimento sobre o domínio, (ii) o conhecimento pedagógico* (o Aprendiz considerado como um agente ativo no processo de aprendizagem), *(iii) o conhecimento sobre o estudante* e *(iv) a capacidade de interação.*

Para alcançar as qualidades discutidas acima, na idealização do modelo [COSTA 97b] objetivou-se uma proposta que abrangesse e viabilizasse, de certo modo, cada uma delas. Tendo estabelecido o modelo de interação mencionado, foi enfatizada, inicialmente, a exigência de conhecimento do domínio (qualidade *(i)*), culminando com uma definição de um modelo multidimensional desse conhecimento. Esta visão multidimensional mostra a possibilidade de um domínio poder ser enfocado por uma visão contextual, sendo que esta visão pode vir acompanhada de várias alternativas de variação do ponto de vista de profundidade e lateralidade em relação a cada contexto escolhido no domínio.

3.3. A Representação de Conhecimento no MATHEMA

Seguindo esta proposta, em [TEIX 97] e no presente trabalho, o modelo MATHEMA para representação do conhecimento do domínio foi utilizado, tendo como metas os seguintes

pontos:

(*i*) a organização do conhecimento utilizando como paradigma a sua visão em dimensões,

(*ii*) a utilização de agentes tutores, responsáveis por ensinar conhecimentos específicos.

A opção pelo MATHEMA implicou na adequação e na distribuição dos conceitos relativos ao domínio da Harmonia Musical através do mapeamento do conhecimento em diferentes visões (dimensões). Esta abordagem constitui uma das características do MATHEMA: a preocupação com qualidade da organização do conhecimento e com os mecanismos de interação entre as entidades no processo de ensino. Portanto, o objetivo principal é um compromisso entre riqueza e estruturação do conhecimento [COSTA 97b].

A visão multidimensional mencionada considera as três dimensões seguintes:

(*i*) o *contexto*, que define múltiplos pontos de vista do domínio,

(*ii*) a *profundidade*, que diz respeito a diferentes níveis de abordagem do domínio e

(*iii*) a *lateralidade*, que está vinculada aos conhecimentos fortemente relacionados ao domínio, tais como os conhecidos pré-requisitos.

Nesta perspectiva, um domínio D qualquer no MATHEMA pode ser definido como o conjunto de vários subdomínios, da seguinte maneira:

$D = \{d_{11}, d_{12}, ..., d_{nm}\}$, onde d_{ij} é definido a partir de um par $<C_i, P_{ij}>$, sendo C_i um contexto particular e P_{ij}, uma profundidade associada a este contexto.

Junto ao contexto e à profundidade atribui-se o conceito de lateralidade, onde para cada subdomínio d_{ij} associa-se uma lateralidade l_{ijk}.

A cada d_{ij} é associado um agente tutor [COSTA 97b], o mesmo acontecendo com cada l_{ijk}. Em resumo, a visão de um determinado conhecimento sobre um domínio é a união de domínios específicos, mais o conhecimento essencial adjacente:

$$d_{11} \cup d_{12} \cup d_{21} \cup d_{22} \cup ... \cup \text{lateralidades correspondentes}$$

Numa visão interna de cada d_{ij}, identificam-se os recursos de ensino: conceitos, resultados, exemplos, contra-exemplos, exercícios resolvidos, problemas, dicas, etc. envolvidos

em cada unidade de conhecimento do *curriculum* em d_{ij} e também em dl_{ij}, onde o l_{ijk} é a lateralidade correlata. Além disso, cada d_{ij} pode ser visto através do desenvolver de diversas situações (*curriculum* ou módulos), juntamente com o referido conjunto de recursos de ensino (Figura 3.2).

No plano superior da Figura 3.2 encontram-se as situações e, logo abaixo, os problemas (com seus relacionamentos intrínsecos) e o material didático necessários às sessões de aprendizagem.

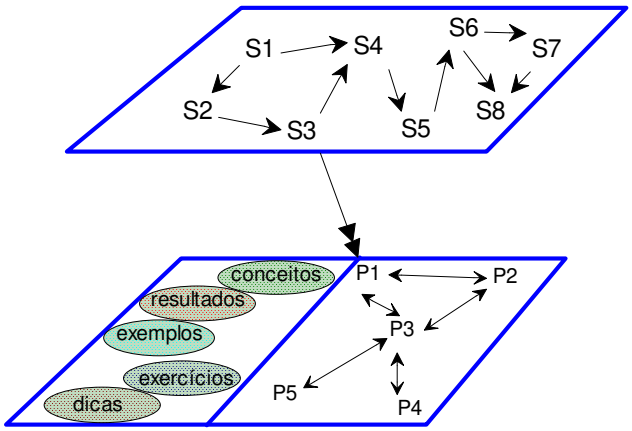


Figura 3.2: visão interna de um agente d_{ij} [COSTA 97b]

Esta visão do conhecimento guarda uma certa semelhança com a estrutura da representação aqui apresentada. Esta semelhança se dá na medida em que, dependendo da profundidade adotada, os conceitos musicais podem estar relacionados a determinados conceitos e ou lateralidades específicas. A Figura 3.3 mostra estas relações, onde um determinado conhecimento pode ser localizado tridimensionalmente de acordo com seu contexto, sua profundidade e sua lateralidade.

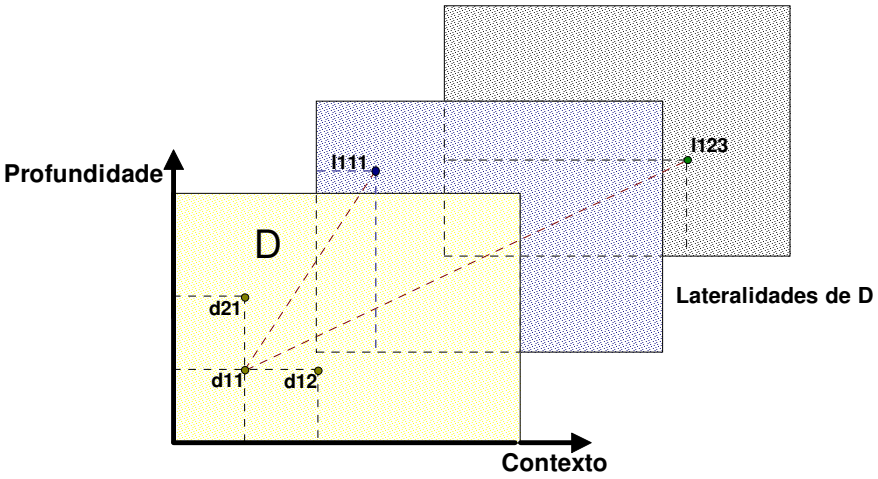


Figura 3.3: dimensões do conhecimento.

3.4. A Representação do Conhecimento em Harmonia Musical

A Harmonia é um exemplo típico de domínio onde se constata a presença de objetos estruturados. Por este motivo, em [TEIX 97], a primeira providência foi identificar quais elementos musicais são preponderantes na resolução de problemas harmônicos. Na primeira fase, a preocupação foi a de isolar as menores partículas musicais e, partindo delas, ir construindo os objetos subseqüentes por ordem de complexidade.

O que se deseja com a representação aqui considerada, é a sua utilização voltada ao ensino de Harmonia. A principal preocupação desta representação é a de prover mecanismos que possam suprir as necessidades de ensino e aprendizagem em Harmonia.

De início, pode-se perceber que as notas musicais, conforme definidas em [TEIX 97], são compostas por freqüências fixas e definidas. Para que estas freqüências sejam utilizadas musicalmente, são necessários alguns conceitos musicais complementares em sua significação. Na Figura 3.4 são mostrados os parâmetros necessários à definição de uma nota, onde os elementos musicais relativos à freqüência estão contidos no retângulo (composto por nome, oitava e alteração).

Por sua composição, a nota possui, em si mesma, uma certa complexidade (encapsulamento), pois a ausência de alguns de seus elementos pode, por exemplo, não ser

relevante em um dado momento e essencial em outro. A parte temporal, como se vê na figura abaixo, fica separada dos atributos de freqüência.

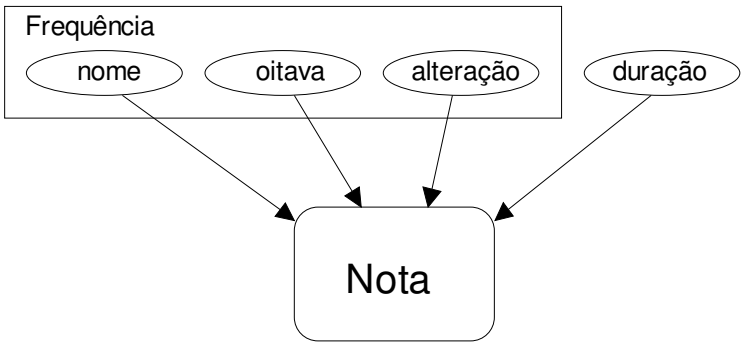


Figura 3.4: atributos de uma nota [TEIX 97]

Com estes componentes é possível situar uma nota qualquer dentro de todo o espectro audível. No caso da pausa, a duração é o único atributo.

Colocando o objeto nota como base das estruturas musicais, tem-se como resultado o fluxograma apresentado na Figura 3.5, com a dimensão das relações entre os vários objetos desse domínio. Aqui a hierarquia intrínseca vai do objeto mais simples (estruturalmente) até o mais complexo.

Seguindo esta ordem encontra-se [TEIX 97]:

- que os *Intervalos* estão em relação de dependência e de herança horizontal com o objeto *Notas*, relação onde o valor de um atributo de um objeto é definido pelo valor deste mesmo atributo em um dos componentes do objeto;
- que as *Escalas* e *Acordes* estão em relação de dependência e herança horizontal com o objeto *Intervalo* e que de acordo com sua disposição (vertical ou horizontal) gera um ou outro,
- que alguns atributos e métodos de *Escalas* e *Acordes* são definidos pelo objeto *Níveis*, que representa os vários estágios de aprendizagem,
- que a relação entre *Escalas* e *Acordes* com o objeto *Encadeamento* se dá através de métodos (regras) de manipulação e tratamento;
- que *Escalas* e *Acordes* definem o objeto *Tonalidade* ao mesmo tempo que são uma abstração desta última e
- que *Tonalidade* define um conjunto singular de *Acordes* e uma única *Escala*.

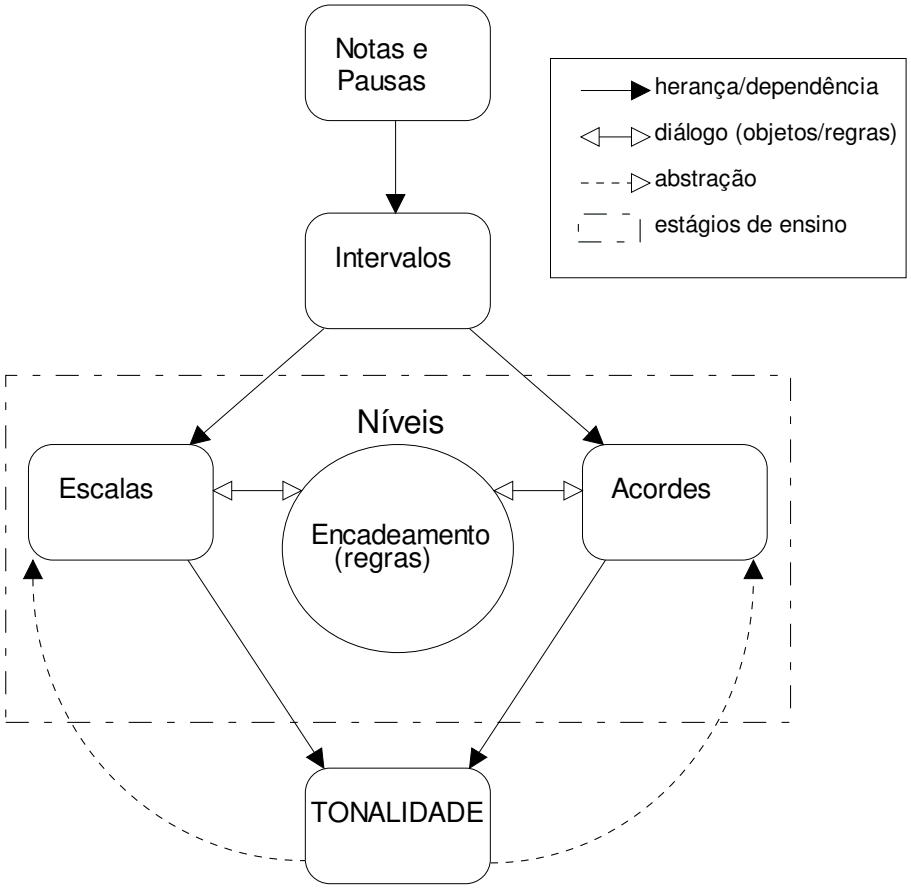


Figura 3.5: estruturação dos objetos musicais relevantes à Harmonia [TEIX 97]

De maneira similar, e principalmente em relação aos acordes, talvez o objeto de maior interesse em Harmonia, a sua estrutura deve permitir certas possibilidades de inferência, exploradas no próximo capítulo, quando estruturarmos os exercícios do nosso tutor, tais como:

- a partir de um acorde incompleto chegar às notas faltantes,
- de um acorde complexo deduzir sua construção básica (sem dissonâncias, etc.),
- construir todas as possibilidades de inversões de um acorde,
- localizar, dentro de uma tonalidade, qual a função de determinado acorde.

Torna-se evidente que para cada um dos exemplos acima, quando tratados por uma representação orientada a objetos, cada variação deve ser vista como objetos independentes. Esta preocupação é necessária pelo fato de que tal complexidade deve ser encarada com um certo desvelo, não só pelo motivo da representação em si, mas, principalmente, pelas necessidades inerentes ao ensino e aprendizagem de tal matéria.

Ainda dentro do objeto *Níveis*, tem-se o objeto *Encadeamento*. Este é responsável pelas regras que irão guiar as diversas possibilidades de resolução de um encadeamento [TEIX 97]. Mais uma vez os mesmos níveis de aprendizado impostos a um aprendiz humano devem ser ressaltados: neste caso, determinadas regras podem ser relevantes em um instante e, em outro, podem ser relegadas a um segundo plano.

Algumas considerações sobre o objeto *Tonalidade*: a *Tonalidade*, em um determinado instante do ensino de Harmonia, possui uma função básica: definir quais acordes e quais escalas farão parte de um problema. De posse dessa informação, pode-se de cara, eliminar uma grande parte do conjunto de acordes e escalas quando da resolução de um exercício.

A Tonalidade serve como limitador de possibilidades em nível didático. Dessa forma, pode-se visualizar as relações entre *Tonalidade*, *Escalas* e *Acordes* da maneira vista na Figura 3.6, onde E é uma escala específica, A's os acordes da tonalidade e C's as cifras implícitas a estes acordes. Aqui, como dito anteriormente, a determinação de uma tonalidade fornece um subconjunto de acordes (e suas cifras respectivas) e uma escala; já a existência de uma determinada escala e/ou acordes nos remete a uma tonalidade específica.

As *cifras* estão associados aos *acordes* e são responsáveis por determinar as funções destes. Dessa maneira, um *cifra* pode estar associada a diversos *acordes*, sendo que a definição biunívoca entre *acordes* e *cifras* é especificada pela *tonalidade*.

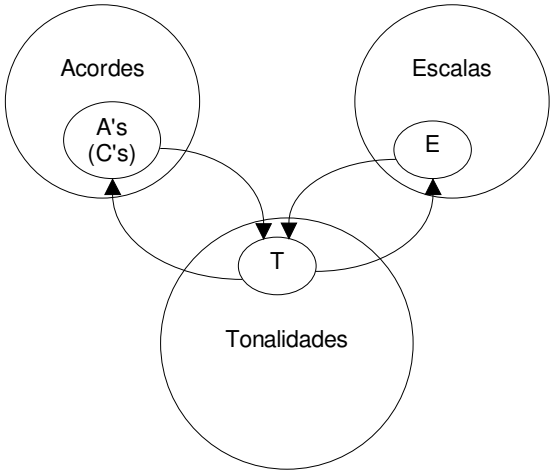


Figura 3.6: relacionamento entre os objetos Tonalidade, Acordes e Escalas [TEIX 97]

Algumas considerações sobre o objeto *encadeamento*: a Figura 3.7 é a ampliação de

encadeamento. Pode-se observar a existência de dois conjuntos distintos de regras que devem ser observadas na resolução de problemas:

- as *regras de encadeamento* propriamente ditas: são aquelas que regem a maneira como devem se relacionar as seqüências de acordes,
- as *regras de distribuição* de vozes: responsáveis por manter cada melodia gerada com as regras de encadeamento dentro de parâmetros estéticos aceitáveis (melodias coerentes, dificuldades de execução por um cantor, extensão e respeitar certos relacionamentos entre as demais vozes).

Como mostra a figura, estes dois conjuntos se inter-relacionam e, com relação à proporção, pode-se dizer que as regras de distribuição são em número menor que as regras de encadeamento. Em [TEIX 97] e ainda no escopo deste trabalho, não se tem a intenção de representar o universo das regras de encadeamento. O motivo é que, de acordo com o encaminhamento dado pelo professor, pode-se acrescentar ou suprimir um conjunto determinado de regras.

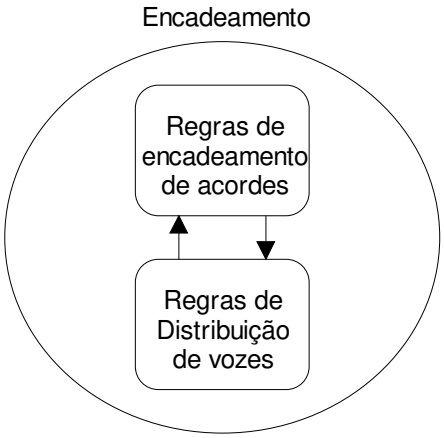


Figura 3.7: objeto encadeamento [TEIX 97]

3.5. Dimensões do Conhecimento em Harmonia

No tópico 3.3, mostrou-se a representação do conhecimento no MATHEMA em um contexto multidimensional, dividido em contexto, profundidade e lateralidade. Explorando agora o domínio da Harmonia, pode-se definir estas três dimensões como um conjunto de elementos, obedecendo a uma organização onde se encontra a definição das diversas fases do ensino (domínios relativos). Entendendo-se o domínio D como sendo o de Harmonia, pode-se considerar, entre outros, os dois contextos seguintes:

C_1 = contexto Harmonia Normativa e

C_2 = contexto Harmonia Funcional.

Cada contexto acima possui níveis de conhecimento (profundidades), desde o mais elementar até o mais complexo. A definição destes níveis garante que o conhecimento seja tratado de maneira gradativa. Por exemplo, ao contexto C_1 podem ser associadas as seguintes profundidades relativas ao conceito de acordes:

P_{11} = acordes maiores, menores, aumentados e diminutos compostos por três notas (terças superpostas) e em posição fundamental,

P_{12} = acordes compostos por quatro notas em posição fundamental,

P_{13} = acordes em posições invertidas (terças no baixo),

P_{14} = acordes com omissão de uma das notas (terça ou quinta),

P_{15} = acorde com omissão de fundamental,

P_{16} = acordes compostos por cinco ou mais notas (9ª, 11ª, 13ª, etc)

P_{17} = acordes com notas estranhas,

P_{18} = acordes dependentes do contexto,

Estes oito exemplos de profundidade podem ser ampliados ou reorganizados de acordo com o objetivo didático almejado pelo especialista.

Uma vez definidos os pares $<C_i, P_{ij}>$, associa-se cada um deles a um subdomínio d_{ij}. Daí, para cada par, deve-se identificar suas lateralidades. A lateralidade pode ser definida também, como os "pré-requisitos" externos que cada d_{ij} necessita para seu entendimento.

Seguindo ainda o exemplo de profundidades do conceito de acordes, têm-se as lateralidades, onde para cada subdomínio d_{ij} associa-se uma lateralidade l_{ijk}. Dessa forma, tem-se a seguinte relação:

l_{111} = conceitos de notas (*classe notas*),

l_{112} = conceitos de intervalos (*classe intervalos*),

l_{113} = conceitos de escalas (*classe escalas*),

No exemplo do conceito de acorde (vide item 2.2.4) existe um:

$$<C_1, P_{11}> = d_{11}, \text{ onde}$$

d_{11} = {conceito de acordes (tríades), exemplos, contra-exemplos, exercícios resolvidos, problemas, dicas} e suas lateralidades

l_{111} = {conceito de nota},

l_{112} = {conceito de intervalo} e

l_{113} = {conceito de escala}.

Os elementos que compõem d_{11} são os seguintes:

- *conceito de acorde*: composto pela *classe acordes* [TEIX 97] e suas variantes,
- *exemplos*: tríades dos acordes formados na escala de mi maior.
- *contra-exemplos*: tríades sem terças superpostas, neste caso inúmeros contra-exemplos podem ser adicionados.
- *exercícios resolvidos*: pode-se fazer a associação entre a representação gráfica do acorde e o seu som ou o símbolo do acorde (B, Gm, E, etc),
- *problemas*: similar ao item acima, sendo que neste caso expõe-se a representação gráfica e pede-se o símbolo ou faz-se o mesmo em sentido inverso, ou apresenta-se acordes incompletos e pede-se o complemento, etc.
- *dicas*: tais como: acordes maiores são formados por uma terça maior e outra menor; acordes menores são formados por uma terça menor e outra maior, etc.

Os itens acima serão discutidos novamente no próximo capítulo, quando definirmos a teoria e os exercícios, problemas e dicas utilizados no tutor em Harmonia.

As lateralidades apresentadas acima estão respectivamente ligadas às *classes nota*, *intervalo* e *escalas*. Vale lembrar que um determinado conceito, dependendo o objetivo, pode ser uma lateralidade ou fazer parte de um subdomínio. No caso da lateralidade l_{111} (conceito de nota), por exemplo, quando este mesmo conceito for tratado como subdomínio, suas lateralidades serão os conceitos de *nome*, *oitava*, *duração* e *alteração*.

O próximo par:

$$<C_1,P_{12}> = d_{12}, \text{ onde}$$

d_{12} = {conceito de acordes (tétrades), exemplos, contra-exemplos, exercícios resolvidos, problemas, dicas} e suas lateralidades

l_{121} = {conceito de nota},

l_{122} = {conceito de intervalo},

l_{123} = {conceito de escala},

l_{124} = {conceito de sétima}.

Os componentes de d_{12} são similares ao d_{11} e à lateralidade pode ser acrescida o conceito de sétima (sensível). Este mesmo esquema deve ocorrer com os outros pares e o conjunto destes que, juntamente com as lateralidades, formam a visão completa do contexto abordado.

Discutiremos no próximo tópico, a representação do conhecimento em Harmonia Musical utilizada neste trabalho, desenvolvida e implementada inicialmente em [TEIX 97].

3.6. O Tutor em LPA-Prolog++

Em [TEIX 97], o ambiente de especificação coincidiu com o de implementação, tendo sido escolhido o LPA-Prolog ++, [MOSS 95] e [VASE 95], linguagem híbrida que une recursos dos paradigmas de orientação a objeto e programação em lógica. Esta escolha se justificou, por um lado, pelo caráter declarativo desta linguagem e, por outro lado, pela necessidade de uma representação hierárquica do conhecimento e de mecanismos capazes de inferir conhecimento a partir de dados incompletos. A opção feita pela Programação em Lógica é devida a sua capacidade de inferir conhecimento a partir de dados incompletos.

Segundo o autor, motivado pela maneira encontrada na organização dos elementos musicais, juntamente com a necessidade de garantir que a manipulação de alguns atributos preponderassem em relação aos demais; optou-se pela utilização do paradigma de Orientação a Objetos (OO). A consistência do paradigma de OO aplicado à Música pode ser avaliada em

diversos trabalhos [POPE 91]. Os benefícios de modularização, encapsulamento e especificação hierárquica de componentes foram preponderantes na escolha da OO. Os elementos musicais foram então organizados em classes que se comunicam entre si (Fig 3.8) para a resolução das consultas feitas pelo Aprendiz ao sistema, em interação textual. A seguir, uma apresentação sucinta destas classes, seus atributos e métodos. O código Prolog completo e adaptado está presente no Anexo 2 deste trabalho e é melhor descrito em [TEIX 97, cap 5].

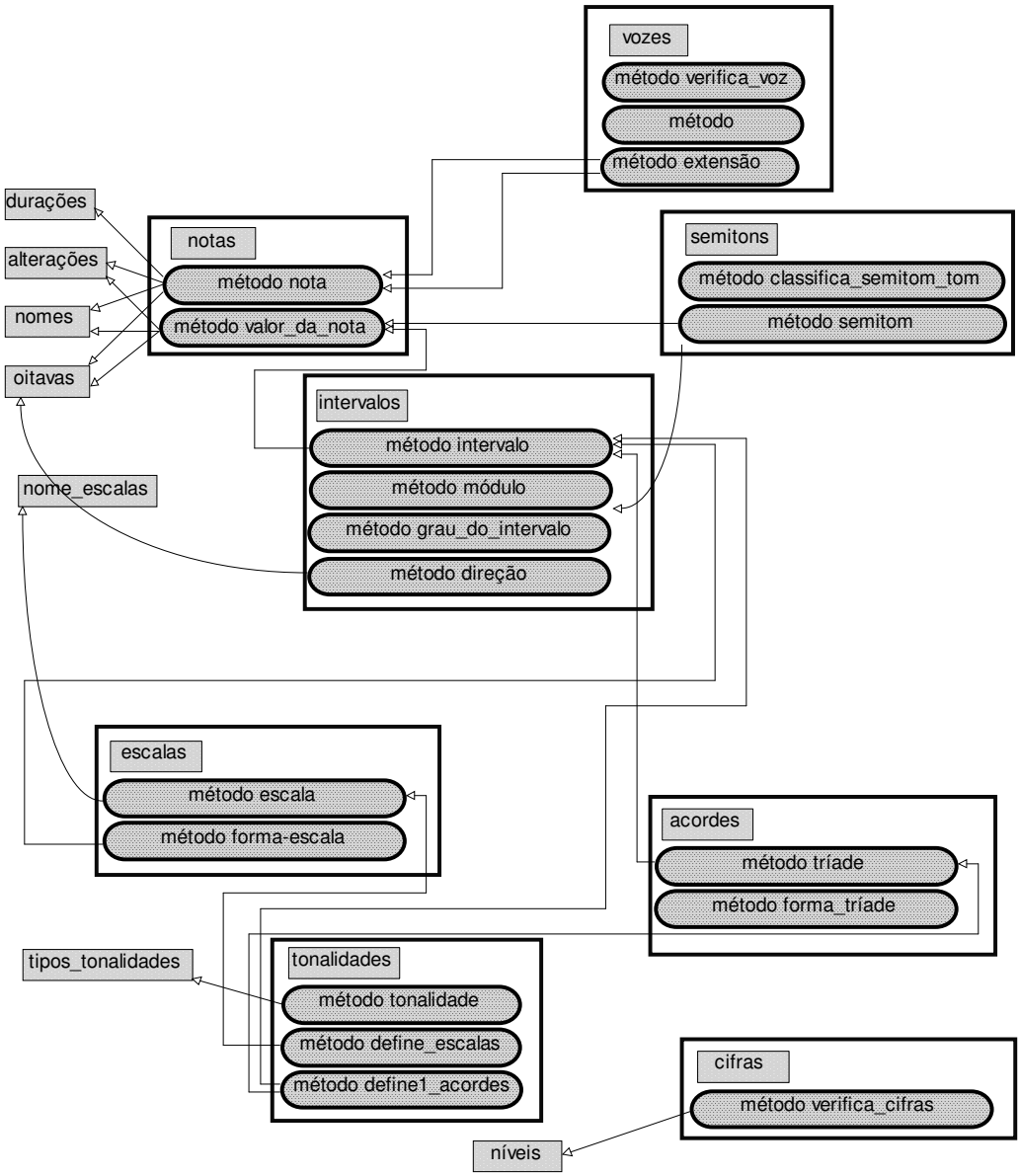


Figura 3.8: diagrama de comunicação entre métodos de classes diferentes [TEIX 97]

3.6.1. Notas e Pausas

Na Figura 3.4 está explícito que toda nota musical é composta por uma lista de parâmetros que necessitam ser nomeados e representados:

- *nome*: baseado na nomeação germânica (a, b, c, d, e, f, g);
- *oitava*: necessário para determinar e situar um nome específico e, conseqüentemente a sua freqüência;
- *duração*: o espaço de tempo em que a nota é executada;
- *alteração*: necessária para a representação do universo dos trinta e cinco possíveis nomes de notas.

Dessa forma, representa-se uma nota como um lista composta por quatro elementos, onde cada um é referência à sua classe de elementos, com os atributos descritos abaixo.

Classe Nomes – possui como declarações a designação dos nomes das notas juntamente com a definição de um valor numérico; o *nome (p)* corresponde às pausas, e os nomes restantes às notas conhecidas, dó (c), ré (d), mi (e), fá (f), sol (g), lá (a), si (b). A cada nome é associado um número que o identifica, tendo como ponto de partida a nota dó (c). Estes valores (1, 3, 5, 6, 8, 10 e 12) são utilizados no método que define o valor final de uma nota. Esta quantificação é feita baseada na disposição espacial das teclas brancas como na Figura 3.9.

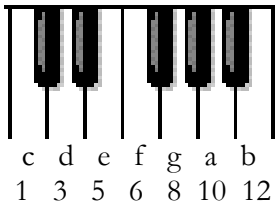


Figura 3.9: valor numérico para nome de notas [TEIX 97]

Classe Oitavas – As oitavas são em número de sete, no entanto, no universo voltado à música coral, apenas quatro oitavas são utilizadas, correspondendo à extensão da voz humana. Na classe oitavas encontra-se a definição de seus nomes juntamente com a definição de um valor numérico (este valor também é utilizado no método de definição do valor final de uma nota).

Classe Durações – A duração de uma nota é definida por um número inteiro qualquer, no entanto, os símbolos musicais para duração já possuem valores numéricos historicamente definidos. Será utilizado um valor *default* (1) para as durações.

Classe Alterações – A *classe alterações* é composta pelo símbolo atribuído a cada uma das alterações e pela atribuição de um valor numérico a cada destes símbolo (natural = 0, sustenido = 1, bemol = -1, dobrado sustenido = 2, dobrado bemol = -2). Para entender as alterações musicais basta associá-las a simples operações matemáticas. As alterações servem para baixar ou elevar semitons e isto é relativo a subtrações e adições feitas nos valores dos nomes das notas.

Classe Notas – A *classe notas* define uma nota como sendo uma lista quádrupla, composta pelas declarações encontrados nas classes anteriores. Dessa maneira, todas as declarações contidas nas classes *oitavas*, *notas*, *alteracoes* e *duracoes* estão agora fazendo parte da classe notas. Esta classe possui um método nota, que é responsável pela verificação de todos os elementos que compõem uma nota, quando da entrada dos dados de oitava, nome, alteração e duração. A partir deste método, já é tratada pela representação a questão da busca de respostas tendo como ponto de partida um conhecimento incompleto. Por exemplo, a ausência de um dos dados de nota é prontamente preenchida pelo método acima, ocorrendo isto com todos os métodos aqui expostos.

O valor de uma nota é produzido por um método que calcula um valor único para cada nota. Este método tem como dados de entrada os atributos de uma nota e retorna um valor numérico para cada uma delas. Este valor é uma representação interna utilizada para a manipulação das estruturas decorrentes do objeto nota. Para cada um dos atributos é feita uma chamada à classe equivalente e é retornado um número. Este número é utilizado para a definição de valores que são computados para a definição exata de valores da nota que são somados com o número 35 para que o retorno do valor numérico seja equivalente ao número MIDI da nota, permitindo a sua reprodução sonora. Estes números são utilizados nos métodos subseqüentes para a definição de outros objetos.

3.6.2. Tom e Semitom

Estes conceitos estão intimamente ligados às definições didático-musicais para acordes e escalas. Pode-se representar o objeto musical tom e semitom através do cálculo da distância entre duas notas. A classe semitons herda horizontalmente as declarações da classe notas e da classe intervalos. Dois métodos privados são responsáveis pela definição da distância entre

duas notas em tom ou semitom; o que for diferente disso é considerado erro. Um exemplo de associação entre teclas e números pode ser visto na Figura 3.10. De acordo com o método, se a diferença entre dois valores de nota for 1, então a distância é de um semitom; se for 2, então a distância é de um tom. Em resultados diferentes a resposta será nula (∅).

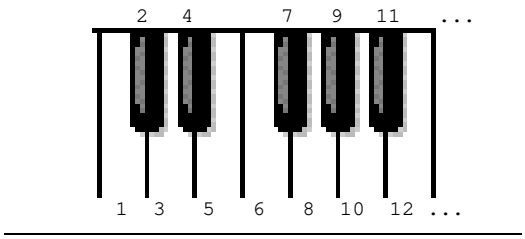


Figura 3.10: valor numérico para notas e suas alterações

3.6.3. Intervalos

A classe tom/semitom encontra-se embutida dentro da classe intervalos. Só que neste caso, os conceitos de tom e semitom assumem denominações diferentes. Já a terminologia utilizada para intervalos em um exercício harmônico é a que aparece abaixo.

Classe Intervalos – herda as declarações da classe notas. Possui um total de quatro métodos, sendo dois privados e dois públicos. Os métodos públicos são responsáveis pela formação do intervalo, como o próprio nome do método sugere, e pela transformação de números negativos em positivos (módulo). Dos métodos privados, um é dedicado ao cálculo do grau do intervalo e o outro a calcular a direção do mesmo.

3.6.4. Escalas

A necessidade de uma classe que defina as possibilidades de escalas, justifica-se pela não existência de 35 escalas diferentes (uma relativa a cada nota com e sem alterações). Na prática, o seu número é reduzido a 30, levando-se em consideração tanto as escalas maiores, quanto as suas relativas menores.

Classe Nome_escalas – é formada por uma lista tríplice, onde o primeiro item é o nome da nota que nomeia a escala, o segundo item é a alteração e, por fim, o modo. No caso do modo não estar diretamente mencionado, subentende-se a existência de ambos.

***Classe Escalas* –** O objeto escala pode ser definido como uma sucessão de intervalos predeterminados, formando em si um estrutura musical coerente. A escala utilizada nos exercícios harmônicos são as escalas diatônicas maiores ou menores (esta última apenas no seu formato harmônico). Por ser composta por oito notas musicais (incluindo a repetição do 1º grau), as escalas diatônicas possuem sete intervalos. Nesta classe, além da definição de heranças e da apresentação dos métodos, existe a declaração de classifica_escala. A lista contida em classifica_escala é responsável pela construção propriamente dita do objeto escala. O primeiro item indica o modo (neste caso o modo maior e modo menor natural e harmônico); o segundo item é a definição de escala diatônica (onde cada nota está na distância de um intervalo de 2º grau); por fim tem-se a lista onde para cada par de notas é definido um intervalo sempre de 2ª, sendo maior, menor ou aumentado, segundo a sua posição na lista.

3.6.5. Acordes

A construção do objeto acordes é bastante similar à construção de escalas. Neste caso, tem-se uma relação de dados voltados à criação de acordes com três e quatro notas (tríades e tétrades).

***Classe Acordes (tríades e tétrades)* –** Após as declarações de herança e relação de métodos, aparece uma lista em classifica_triade nos mesmos moldes da encontrada em classifica_escala. O primeiro item diz respeito ao tipo da tríade/tétrade, o segundo ao grau dos intervalos (3º grau, ou seja, terças), a relação de intervalos entre notas e o quantificador (2 no caso das tríades e 3 no caso das tétrades). O método tríade é responsável pela classificação do acorde. Este é um método típico que pode ser aplicado ao ensino, pois de acordo com o nível que se deseja trabalhar pode-se ampliar a base de dados contida em classifica_triade, incluindo outros tipos de tríades/tétrades.

Em ambos os métodos para a formação de escalas e acordes, a inferência pode ser feita não apenas partindo de uma nota e uma descrição; mas a partir de um acorde, completo ou incompleto, tem-se condições de definir o acorde e de encontrar as notas faltantes.

3.6.6. Tonalidade

A Tonalidade é responsável pela definição de determinadas relações entre a notas.

Estas relações são de grande importância no ensino de Harmonia. Na prática, conforme a Figura 3.6, pode-se ver a tonalidade como a definidora de quais acordes e escala serão utilizados em um determinado instante. Portanto, ao objeto musical Tonalidade, têm-se duas classes distintas: a que define os tipos de acordes possíveis em um determinado nível; e a que é responsável pela relação tonalidade-escala-acorde. Além da capacidade de abstrair certos acordes de seu conjunto total, a classe tonalidades é que define a função destes acordes. Como exemplo têm-se os principais acordes que são montados sobre o I, IV e V graus da escala (vide cap 4) e esta, com dito acima, é definida pela Tonalidade. Por este motivo, considera-se a Tonalidade como um objeto distinto, pois a sua presença em uma sentença lógica será responsável pela limitação de acordes e escala. Portanto, o código de Tonalidade pode ser visto de duas maneiras:

- *Definição de escala e acordes*: entra-se com os dados sobre uma determinada tonalidade (nome, alteração e modo) e a saída complementa os dados definindo a escala e os acordes.

- *Definição de relações entre acordes*: onde é feita a distribuição em graus (associação de cada acorde gerado com os graus da escala). A importância de um método responsável por esta ação é bem visível quando da análise e correção de exercícios, onde uma visão geral pode ser feita antes das considerações sobre o encadeamento

Classe Tipos_Tonalidades – A *classe tipos_tonalidades* é responsável pela nomeação dos tipos de tonalidades necessárias. Caso se deseje um universo menor de tonalidades basta restringir o número de declarações desta classe. É descrito em *tipo_tonalidade*, todas as tonalidades possíveis (15 tonalidades maiores e 15 tonalidade menores).

Classe Tonalidades – Esta classe é responsável pela construção das tonalidades, seja através da entrada de todos os seus dados para validação (incluindo os acordes e escala), seja através da saída de alguns de seus objetos componentes.

Como coadjuvante da *classe tonalidades* temos a *classe níveis* e a *classe cifras* que serão vistas a seguir.

Classe Cifras – A classe cifras é composta apenas pelo método que faz a verificação se as cifras dadas como resposta pelo aluno estão contidas dentro da tonalidade e do nível

pretendidos. O método para verificação de cifras faz a chamada à classe niveis, verificando a existência do acorde em um nível específico.

Classe Níveis **–** Aqui, de acordo com os objetivos do professor-especialista, pode-se acrescentar quantos níveis forem desejados.

3.7. Conclusões

O presente trabalho está inserido no contexto da Inteligência Artificial por utilizar técnicas de representação do conhecimento além de outras técnicas da IA para a modelagem e funcionamento do sistema. Para a modelagem utilizou-se o modelo MATHEMA para construção de STIs que utiliza uma representação do conhecimento baseada em dimensões e lateralidades, usada na construção da Base de Conhecimento (ATs) e no Módulo Tutorial presentes no ambiente proposto aqui. A Base de Conhecimento é responsável por prover respostas e estruturas corretas durante a interação de tutoramento com o Aprendiz. Este módulo foi implementado em LPA-Prolog++, por suas características declarativas e que possibilitam inferências necessárias neste tipo de ambiente, e comunica-se com os outros módulos através do Agente Comunicador. Todos os outros módulos do sistema foram implementados em JAVA, para prover um ambiente que pode ser utilizado via Internet, que agrega características de sistemas multimídia e com potencial para programas de tutoramento à distância.

O objetivo é utilizar um esboço de representação em harmonia musical como base de conhecimento para um sistema tutor inteligente, através do modelo MATHEMA e suas características de representação do conhecimento e tutoramento, gerando um ambiente de aprendizagem em harmonia musical, multimídia, disponível via Internet e para instalação em escolas de música como também no ambiente de estudo dos aprendizes, suas casas ou laboratórios para exercícios.

Neste capítulo abordamos o ambiente MATHEMA desenvolvido em [COST 97], seu modelo para STIs multi-agentes, a representação do conhecimento em harmonia musical segundo este modelo, seus níveis e lateralidades, o esboço da base de conhecimento em Harmonia Musical construída em [TEIX 97], suas características e divisão por classes, que

foram utilizados no desenvolvimento do ambiente proposto aqui, com sua modelagem, especificação formal e implementação abordadas no capítulo 5.

No próximo capítulo será abordada a Harmonia Musical, área do conhecimento escolhida para a elaboração deste trabalho, onde serão explorados aspectos relacionados ao aprendizado deste domínio bem como o formato das interações Aprendiz x Tutor, o formato utilizado nas explanações teóricas sobre cada tópico do domínio, exemplos de exercícios e dicas, utilizados no desenvolvimento do ambiente proposto neste trabalho.

Capítulo 4
Harmonia Musical

4.1. Introdução

No desenvolvimento de sistemas baseados em domínios do conhecimento como a Harmonia Musical, um dos maiores problemas encontrados é: como representar computacionalmente as estruturas musicais?. Neste domínio temos que considerar dois grupos distintos de objetos musicais: o formal, composto por objetos como notas, tempo, tonalidades, alterações, etc., e o cognitivo, voltado ao aspecto perceptivo, com objetos como tensão, relaxamento, emoção, contexto histórico e performance. Este trabalho utiliza, como solução, o modelo de representação e manipulação do conhecimento musical citado no capítulo anterior.

Outro problema encontrado em se tratando de sistemas educacionais musicais é em relação à comunicação entre o Aprendiz e o Sistema, de forma que contemple as necessidades desta interação no tocante à facilidade de manipulação do sistema, compreensão do conteúdo educacional apresentado, máxima apreensão da atenção e dos sentidos do usuário com a variedade nas formas utilizadas para apresentação deste conteúdo e captação das respostas do Aprendiz. Neste tipo de sistema, os aspectos sonoros e gráficos são fundamentais para uma total compreensão dos tópicos abordados. O presente trabalho apresenta como solução, o desenvolvimento de um sistema tutor que comunica-se através de uma interface multimídia, tendo como base o modelo MATHEMA.

Neste capítulo são abordados ainda conceitos da Harmonia Musical, um esboço do material utilizado no presente trabalho para o tutoramento destes conceitos e exemplos de interações e exercícios.

4.2. O que é Harmonia ?

A Harmonia tem na música várias definições como: "uma sucessão de acordes", "o encadeamento dos sons simultâneos", "o resultado do encontro vertical das vozes do contraponto" ou "parte da tecnologia musical que trata da simultaneidade dos sons".

Segundo [TEIX 97], ao se aprofundar no universo musical descobre-se que a compreensão pelo conceito para Harmonia encontrado no Aurélio, "agradável sucessão de sons", acaba se tornando uma ciência com grande lastro de estudo. Isto se dá pelas inúmeras possibilidades e maneiras de enfocar este assunto que, por sua vez, se reflete em muitas teorias que culminam na criação musical propriamente dita. E é a isto que a Música, e em particular a Harmonia, vem se sujeitando nos últimos séculos.

Em [SCHO 79], é esta visão, a de uma ciência Harmônica, que se persegue como modelo. O ensino de Harmonia, deve conscientizar o aluno sem a cobrança de torná-lo um compositor. Já para um compositor, ela deverá servir de suporte à compreensão, criação e execução.

4.2.1. Harmonia Funcional

Em [BRIS 79], "o conceito de função surgido evidentemente muito mais tarde, na história da música, decorre da idéia de estudar esses acordes quando em movimento, isto é, na concatenação harmônica, e as relações que mantém entre si e com o todo, nesse movimento".

Na Harmonia Funcional, cada acorde assume um função dentro do universo harmônico. Estas funções são determinadas pela "atração" que cada acorde exerce uns em relação aos outros. Assim sendo, os acordes de primeiro, quarto e quinto graus recebem as funções de tônica, subdominante e dominante, respectivamente. Todo um estudo é feito para corroborar estas funções. Em torno delas irá girar todos os outros acordes.

Os acordes são identificados através das iniciais de tônica, subdominante e dominante: "T", "S" e "D", e como todo o restante dos acordes são relacionados com estes três, para as tonalidades relativas e anti-relativas (menores e maiores) acrescentasse a letra "r" ou "a" respectivamente: Tr, Sr, Dr, Ta, Sa e Da. Isto implica em se ter apenas três símbolos (e suas variações) para todos os acordes encontrados em uma peça musical.

4.2.1. Harmonia Normativa (Tradicional)

A Harmonia Normativa trabalha com regras de condução das vozes. Estas regras por sua vez, possuem determinados graus de importância, segundo o estágio em que se encontra o aluno. No início, este se vê envolto em algumas poucas regras, destas ele não deve fugir. Como conseqüência do acúmulo de conhecimento, novas regras são, aos poucos, dispensadas enquanto outras vão surgindo. A Harmonia Normativa utiliza os algarismos romanos como escrita para a representação dos graus da escala. No ensino da Harmonia, através da escola Normativa, as regras de encadeamento são acrescentadas para o aluno paulatinamente. Estas regras são responsáveis por prender o aluno ao que estiver sendo prioritariamente ensinado. [TEIX 97]

Abordaremos no próximo tópico os conceitos da Harmonia e da Teoria Musical utilizados neste trabalho. Estes conceitos e definições estão detalhados em [CHED 85], [HIND 86], [PRIO 85], [SCHO 79] e outros.

4.3. Conceitos Fundamentais da Harmonia Musical

Os conceitos dos objetos musicais descritos aqui foram usados como conhecimento musical no momento da representação do conhecimento descrita no capítulo 3 deste trabalho. Aqui, serão apresentados com a intenção de se definir o conteúdo a ser apresentado durante o tutoramento, transformados em teoria, exercícios, dicas, etc., itens trabalhados no próximo tópico.

Ao examinar uma partitura, pode-se catalogar todos os seus elementos formais. Estes elementos são a matéria-prima com a qual a maioria dos teóricos trabalham. Cabe ressaltar as estruturas musicais consideradas complexas (como acordes, escalas e tonalidades), não só por suas constituições (formada por estruturas mais simples) mas também pelo inter-relacionamento destas estruturas dentro de determinados contextos. Por exemplo, as diversas funções (tônica, dominante, relativa da dominante, etc) que um mesmo acorde pode assumir em tonalidades diferentes.

O pentagrama é um conjunto de cinco linhas (figura 4.1) e quatro espaços (também conhecida como pauta) sobre os quais se escrevem as figuras musicais (notas, pausas, símbolos, etc). Um pentagrama possui no seu início uma clave (que determina o nome das notas), uma armadura (indica a tonalidade) e um compasso (métrica) definidos respectivamente pelos três primeiros símbolos [PERE 85].

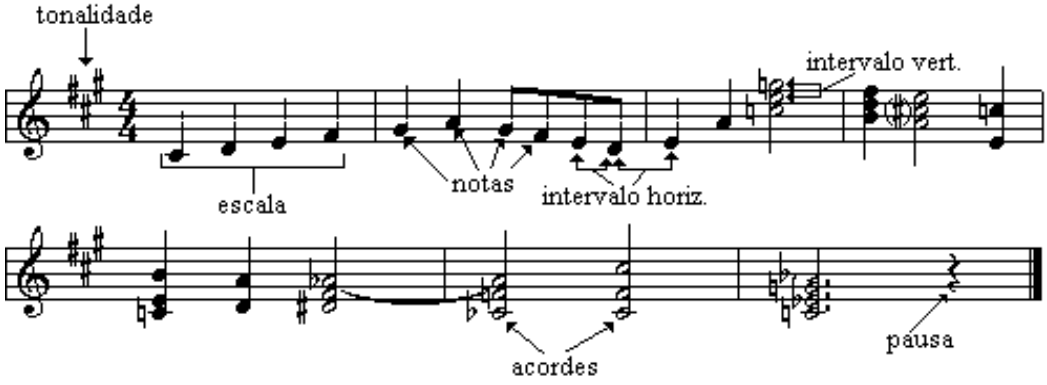


Figura 4.1: objetos musicais em pentagrama [TEIX 97]

O símbolo de som é denominado nota e o símbolo do silêncio é denominado pausa. Intrinsecamente, cada um destes símbolos traz uma série de atributos necessários ao som. Portanto, pode-se dizer que notas e pausas, em analogia à Física, constituem os "átomos" da Música. Com um conjunto ordenado destes átomos, podem-se formar diversas estruturas musicais como os intervalos, formados quando da existência de duas notas, as escalas, conjuntos de intervalos, dispostos melodicamente (sons consecutivos) dentro de determinados padrões intervalares (distâncias), os acordes, conjuntos de intervalos verticais (três ou mais notas), dispostos segundo as leis da Harmonia.

4.3.1. Notas

A freqüência sonora da nota é definida pelos atributos *nome*, *oitava* e *alteração*. Mas para a definição de uma nota, faz-se necessário a noção de mais um atributo que é a sua *duração*.

Nome

Os nomes das sete notas musicais naturais são: dó, ré, mi, fá, sol, lá e si. Um outro tipo de convenção se utiliza de letras para representar as notas musicais. Este tipo de notação é

chamada de notação alfabética e remonta aos gregos. Este tipo de notação é utilizado nos países anglo-saxões (seu outro nome é sistema germânico de notação), em tratados de física. Sua difusão foi bastante acentuada por sua utilização na música popular (cifras), como forma de abreviar e economizar espaço [PERE 85]. Esta é sua equivalência:

dó (c), ré (d), mi (e), fá (f), sol (g), lá (a), si (b)

Oitava

Oitava é um símbolo de execução que resulta em dobrar (oitava alta) ou dividir (oitava baixa) a freqüência de uma determinada nota. Na Figura 4.2, pode-se ver a disposição das oitavas em um teclado de piano. O asterisco indica, no teclado, o início de cada oitava (nota dó).

Figura 4.2: teclado e oitavas

Alteração

Alterações são símbolos utilizados para elevar ou abaixar uma nota, um semitom ou um tom a partir de sua afinação original. O sustenido (#), eleva um semitom, o dobrado-sustenido (**x**), eleva um tom, o bemol (*b*) que abaixa um semitom e o dobrado-bemol que abaixa um tom (*bb*). Em determinados casos as notas podem vir acompanhadas de uma dessas alterações, se o desejo for o de retornar a nota ao seu estado original (teclas brancas), utiliza-se o símbolo do bequadro (♮).

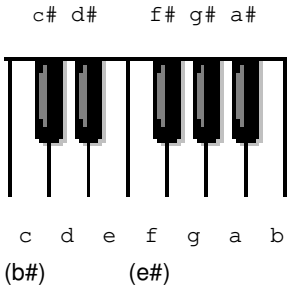


Figura 4.3: naturais e sustenidos [TEIX 97]

Como conseqüência imediata da colocação das alterações, o número possível de notas passa de sete para trinta e cinco, sendo sete naturais, sete sustenizadas, sete bemolizadas, sete com dobrado-sustenido e sete com dobrado-bemol. Na oitava de um Piano ou Teclado, algumas teclas são referidas com mais de um nome (são doze teclas para trinta e cinco nomes), isto se deve ao que se chama de temperamento da escala musical. Este temperamento gerou o conceito de enarmonia.

Enarmonia

Atualmente, os sons ou intervalos enarmônicos são os que, embora tenham nomes diferentes, são iguais em altura (freqüência) dentro do sistema temperado. Em alguns instrumentos desprovidos de teclado, o sustenido é mais elevado (freqüência um pouco mais alta) que o bemol [PERE 85].

Duração

Em Música, a duração das notas obedece uma proporcionalidade e uma relatividade entre os símbolos que representam o tempo. Não se pode afirmar categoricamente que uma determinada figura musical possua um valor de tempo que seja imutável. O parâmetro para este tipo de determinação é a fração colocada no início de um pentagrama (Figura 4.1). O numerador diz quantos tempos possui a *métrica* da música (por exemplo: 1, 2, 3, 1, 2, 3, ... ou 1, 2, 3, 4, 1, 2, 2, 3, 4, ..., etc). O conceito de métrica está intimamente relacionado com o conceito de compasso que é uma divisão rítmica regular, representada graficamente por barras verticais (Figura 4.1) e está vinculado com a fração do início do pentagrama. Já o denominador especifica qual a figura musical que equivale à pulsação da música. A Figura 4.4 mostra a numeração de cada figura. Uma figura logo abaixo da utilizada como referencial terá a metade da duração da anterior e a que lhe antecede, o dobro.

Todas estas informações estão contidas em qualquer partitura musical. A duração e o ritmo dependem da figura de tempo utilizada. A distribuição espacial (sentido horizontal) destas figuras, sobre um pentagrama, indica o ordem cronológica dos acontecimentos. Já a altura das notas (freqüências) está relacionada com a disposição destas no sentido vertical. Quanto mais em cima, em relação ao pentagrama, mais aumenta a freqüência, o contrário

também é verdadeiro: se a nota é colocada mais para baixo, diminui a freqüência.

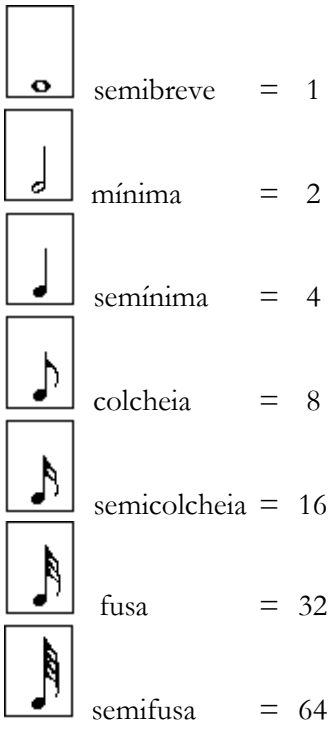


Figura 4.4: representação gráfica e numérica de figuras de tempo [TEIX 97]

4.3.2. Semitom e Tom

Estes dois conceitos e os próximos a serem apresentados, passam a ser obtidos a partir da consideração de duas notas subseqüentes. Estes conceitos trabalham com duas ou mais notas (intervalo) e levam em consideração a distância entre elas (distância entre freqüências).

Semitom

Um semitom, tendo como base um teclado de piano, é a distância entre teclas vizinhas, sejam estas brancas ou pretas. Na Figuras 4.3, pode-se ver alguns exemplos de semitom: entre c-c#, entre e-f, etc. No caso do intervalo (c-c#), este tipo de semitom é chamado de semitom cromático, por ocorrer entre notas com o mesmo nome (c, no caso) sendo que uma ou ambas devem possuir uma alteração. Se ocorrer entre notas de nomes diferentes (e-f) é chamado semitom diatônico.

Tom

Para o conceito de tom, pode-se encontrar diversas acepções. De uma maneira prática, pode-se afirmar que tom é a distância entre teclas, formada por dois semitons. Por exemplo: c-d, e-f#, etc.

4.3.3. Intervalo

Intervalo é a distância entre duas notas. Quando as notas que compõem o intervalo se encontram no perímetro de uma oitava, tem-se um intervalo simples e, quando ultrapassa uma oitava, tem-se um intervalo composto.

Um intervalo possui como características:

- **natureza** (vertical - horizontal): se o intervalo ocorrer em uma melodia, estará caracterizada a natureza horizontal do intervalo (melódico). Caso ocorra entre notas de melodias diferentes e simultâneas, estará caracterizado o intervalo vertical (harmônico).

- **direção**: de acordo com a posição da segunda nota, o intervalo pode ser ascendente (quando a segunda nota é mais aguda que a primeira), ou descendente (quando a segunda nota é mais grave que a primeira).

- **grau**: o grau do intervalo é medido pela distância que separa dois nomes de notas dentro da lista (c, d, e, f, g, a, b), repetindo-se em todas as oitavas do teclado. Os intervalos acontecem entre quaisquer nomes e a sua contagem deve incluir o primeiro nome, os nomes compreendidos entre este e o segundo nome. O resultado é o nome do grau. A Figura 4.5 exemplifica todos os graus de um intervalo simples tendo como base a nota c.

Número de Semitons: dependendo do grau do intervalo, o número de semitons irá estabelecer o restante do nome do intervalo (diminutos, menores, maiores, aumentados e justos). A Tabela 4.1 dá a nomeação dos graus.

Para a definição completa do nome de um intervalo, são necessárias três partes distintas:

i. a que diz qual a direção do intervalo, se ascendente ou descendente;
ii. a parte composta pelos nomes das notas que compõem o intervalo, o que determina o seu grau; e
iii. a que contém o número de semitons entre as notas. Estas partes, unidas, resultam no nome completo do intervalo, que será, por exemplo: intervalo de 2º menor ascendente e assim por diante.

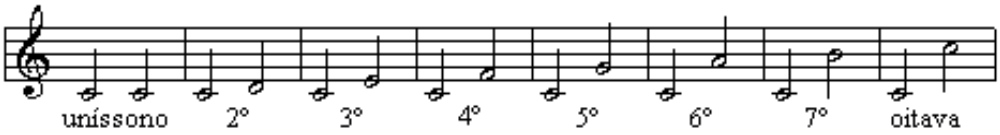


Figura 4.5: graus de intervalo [TEIX 97]

Tabela 4.1: nomes dos intervalos e seus graus

	nº de semitons				
grau	DIMINUTO	MENOR	JUSTO	MAIOR	AUMENTADO
2ª	0	1		2	3
3ª	2	3		4	5
4ª	4		5		6
5ª	6		7		8
6ª	7	8		9	10
7ª	9	10		11	12
8ª	11		12		13

Tabela 4.2: inversão de intervalos

Intervalo	Invertido
uníssono	oitava
2ª	7ª
3ª	6ª
4ª	5ª
5ª	4ª
6ª	3ª
7ª	2ª
oitava	uníssono

Inversão de intervalos

Uma das propriedades de um intervalo diz respeito à sua inversão. Inverter um intervalo é o mesmo que deslocar a nota mais grave uma oitava acima. Desta maneira, para um determinado intervalo simples existirá um determinado intervalo invertido (Tabela 4.2).

Da mesma maneira, pode-se dizer, com respeito ao número de semitons entre os intervalos, que: intervalos maiores, quando invertidos, transformam-se em menores e vice-versa, intervalos diminutos transformam-se em aumentados e vice-versa e intervalos justos continuam justos.

4.3.4. Escala

Escala é uma "sucessão ascendente e descendente de vários sons, com intervalos que guardam entre si uma certa relação de tons e semitons" [PERE 85]. Existem vários tipos de escalas. Neste trabalho serão consideradas as escalas diatônicas, que correspondem à seqüência natural de tons e semitons. As escalas diatônicas eqüivalem à sucessão de oito notas obedecendo ao esquema da Tabela 4.3.

É necessário observarmos que a escala utilizada atualmente na música ocidental é a chamada escala temperada. Esta se divide em 12 semintons iguais, eliminando dessa forma a necessidade de se ter um instrumento afinado com cada uma das escalas possíveis. No temperamento, as 35 notas vistas no subitem 4.3.1, são confinadas em doze dando uma frequência fixa (gerando enarmonias). Na escala acústica natural há diferenças físicas (freqüência) entre duas notas com nomes iguais. Por exemplo: um dó sustenido soará uma freqüência um pouco mais alta que o ré bemol.

Como variações da escala diatônica, têm-se duas seqüências de tons e semitons: modo maior e modo menor. O modo maior corresponde à Tabela 4.3. O modo menor possui algumas variações possíveis. Na Tabela 4.4 são mostradas as escalas menores em suas formas natural, melódica e harmônica sendo esta última a mais utilizada no ensino de Harmonia.

Tabela 4.3: distância entre os graus de uma escala diatônica.

Graus	Distância
entre o 1º e 2ª graus	distância de um tom
entre o 2ª e 3ª graus	distância de um tom
entre o 3ª e 4ª graus	distância de um semitom
entre o 4ª e 5ª graus	distância de um tom
entre o 5ª e 6ª graus	distância de um tom
entre o 6ª e 7ª graus	distância de um tom
entre o 7ª e 8ª graus	distância de um semitom

Tabela 4.4: distância entre os graus das escala diatônica menores.

Grau	Natural	Harmônica	Melódica
1º e 2ª graus	tom	tom	tom
2ª e 3ª graus	semitom	semitom	semitom
3ª e 4ª graus	tom	tom	tom
4ª e 5ª graus	tom	tom	tom
5ª e 6ª graus	semitom	tom	semitom
6ª e 7ª graus	tom	tom + semitom	tom
7ª e 8ª graus	tom	semitom	semitom

4.3.5. Acorde

Um acorde é a emissão simultânea (intervalos verticais) de três ou mais notas, dispostas segundo as leis da Harmonia. Os acordes podem ser *consonantes* ou *dissonantes*, *maiores* ou *menores*, *tríades*, *tétrades*, de *sétima*, etc.

Após toda a revolução provocada pela Música Contemporânea, tem-se hoje um universo bastante amplo de acordes possíveis. Ao se limitar, por exemplo, às primeiras lições da Harmonia, reduz-se o número de acordes de interesse. Em um estágio inicial, apenas os acordes maiores e menores são trabalhados. Posteriormente, passa-se a trabalhar com os acordes de sétima, diminutos, e aumentados.

Em relação à escala, os acordes utilizam cada uma das notas da escala para a formação de um conjunto de acordes da seguinte maneira: cada nota da escala dará nome a um acorde

específico (e suas variações), segundo descrição abaixo, formando assim um conjunto de sete nomes possíveis para os acordes.

Similarmente às escalas, os acordes são formados segundo determinadas relações intervalares. Para resumir a construção de um acorde pode-se afirmar:

- acordes *maiores, menores, diminutos* e *aumentados* são formados com três notas (tríades);
- acordes de *sétima* são formados com quatro notas (tétrade);
- todo acorde possui uma nota que lhe dá nome (inicialmente trabalha-se com acordes em posição de fundamental, ou seja, a nota que dá nome ao acorde é sempre a mais grave);
- partindo da fundamental (inferior), tem-se uma distância de 3º grau até a segunda nota (chamada de terça do acorde) e de 5º grau até a terceira nota (chamada de quinta do acorde). Por exemplo: partindo da nota C tem-se como 3º grau a nota E e como 5º grau a nota G (terças superpostas), formando o acorde de dó. Tratando-se de acordes de sétima, tem-se uma quarta nota posta a distância de 3º grau da quinta do acorde. Por exemplo, o acorde de dó com sétima terá a nota b acrescentada às anteriores.

Uma vez definidas as notas que compõem o acorde, resta definir o tipo do acorde. A Tabela 4.5 fornece os intervalos entre as notas do acorde e a denominação do mesmo.

Tabela 4.5: relação entre terças para formação de acordes.

Tipo	3ª inferior	3ª superior	sétima
maior	maior	menor	–
menor	menor	maior	–
diminuto	menor	menor	–
aumentado	maior	maior	–
com sétima	maior	menor	menor

Um acorde, em determinados momentos, pode aparecer desfigurado: vir desfalcado de uma de suas notas (acorde incompleto), ser acrescido de outras notas que não façam parte do mesmo (notas de passagem), etc.

4.3.6. Cifras

Cifras são símbolos que identificam relações entre notas dentro de um acorde. As cifras podem ser empregadas para diversos fins, aqui ela é utilizada para a definição dos graus de uma escala e, posteriormente, na disposição dos acordes. Por exemplo, na escala de dó cada nota é identificada por um algarismo romano, nomeando dessa forma os graus da escala, como se vê abaixo.

`I` (`c`), `II` (`d`), `III` (`e`), `IV` (`f`), `V` (`g`), `VI` (`a`), `VII` (`b`)

4.3.7. Tonalidade

Tonalidade é o denominador do caráter de uma obra musical. Quando se sabe a tonalidade de uma obra, sabe-se também as principais relações entre as notas que a compõem. A tonalidade indica quais os principais acordes e qual a escala utilizada.

Atualmente, o conceito de tonalidade engloba também o conceito de *modo*. O modo, em Música, acaba se misturando com o conceito de escala. Existem escalas maiores e menores, assim como tonalidades no modo maior e menor. Isto já é bastante corriqueiro, embora alguns teóricos façam distinção entre as duas coisas: tonalidade e modo [CARV 94]. O que deve ficar claro é que: *(i)* tonalidades maiores se apoiam em escalas maiores e *(ii)* tonalidades menores se apoia em escalas menores (sempre escalas menores harmônicas, para efeitos didáticos).

Todas as tonalidades maiores possuem uma tonalidade relativa no conjunto de tonalidades menores. Por exemplo: a tonalidade de *lá menor* é a relativa da tonalidade de *dó maior*. Uma maneira de se determinar uma tonalidade, ou sua relativa, é através das alterações postas no início de um pentagrama (Figura 4.1), como também através da audição e/ou de determinadas características que facilitam a definição dos modos.

4.4. O Tutoramento no MHITS

Uma das principais contribuições deste trabalho é a parte específica da interface onde acontece o tutoramento. O tutoramento utiliza uma estratégia pedagógica fixa e acontece nos

níveis de aprendizagem disponíveis no sistema (iniciante, intermediário, avançado), através da apresentação de uma teoria inicial de cada nível, de uma teoria específica para cada exercício, de dicas para cada exercício e exemplos. A interface é baseada em um sistema hipermídia, implementada utilizando tecnologia Internet, possibilitando a sua utilização em programas de educação à distância. O capítulo 5 trata dos aspectos de especificação e desenvolvimento do sistema, com tópicos dedicados à interface.

O MHITS não se prende ao modelo tradicional Teoria/Exemplo/Exercício/Avaliação, ele disponibiliza formas de interação Aprendiz-Tutor para o enriquecimento da relação ensino-aprendizagem. No MHITS, os tutoriais de cada nível seguem uma ordem predefinida mas com possibilidades de consulta ao sistema pelo Aprendiz através de dicas, exemplos, perguntas, dúvidas, todas disponíveis a partir da interface do sistema. Além destes recursos de interação é livre para o Aprendiz o contato com o responsável humano (SEH, cap. 3) através de serviços Internet como e-mail, salas de bate-papo, listas de discussão, implementados especificamente para a aplicação do sistema em cursos a distância.

O desempenho do Aprendiz em cada tutoramento é avaliado através da pontuação adquirida e armazenada pelo sistema. Os passos da interação do Aprendiz em cada nível do sistema também são armazenados para fins de avaliação e avanço de nível.

Cada exercício possui uma pontuação e um peso próprios, com tônica ou notas diferenciadas para cada exercício, permitindo a inclusão de uma opção para repetição do exercício sem uma repetição obrigatória da nota, tônica ou estrutura específica. No final de cada nível ou de cada interação é calculada a média alcançada pelo Aprendiz. Esta é uma forma de avaliação que dá a liberdade ao Aprendiz de repetir exercícios conforme a sua necessidade de aprendizagem de determinado conceito. Está prevista a implementação de um módulo administrativo que permite ao professor (SEH) decidir que exercícios serão apresentados e com quantas repetições para cada interação em cada nível do sistema.

Os subtópicos a seguir apresentam esboços das seqüências de telas que compõem cada exercício, baseados nos conceitos musicais e de Harmonia apresentados acima. Cada nível possui uma quantidade específica de exercícios, com a exposição de uma teoria inicial relativa aos tópicos explorados naquele nível, e cada exercício segue basicamente o mesmo modelo de apresentação, idealizado pelo autor seguindo os modelos pedagógicos tradicionais: *(i)*

apresentação da teoria, *(ii)* apresentação dos exercícios e estruturas interativas que possibilitam a resposta do Aprendiz, *(iii)* a possibilidade de acesso a uma ou mais dicas para o exercício, *(iv)* tela de avaliação do sistema com possibilidade de escolha para repetição ou avanço para outro exercício, ou avaliação final do nível, caso seja o último exercício previsto.

Esta forma de interação e apresentação do material tutorial possibilita uma modelagem segundo o MATHEMA, com a pretensão de atingir as dimensões e lateralidades do domínio do conhecimento mais importantes para cada nível e interação, através da escolha, pelo professor, dos exercícios que compõem cada um destes.

No capítulo 5 será apresentada a especificação e modelagem do sistema, como também a forma de comunicação entre a interface e a Sociedade de Tutores (SATA), a comunicação entre eles para a avaliação da resposta do Aprendiz. O sistema prevê ainda um Módulo de Perguntas, com acesso disponível a partir de qualquer tela no tutorial, que dá ao Aprendiz a possibilidade de fazer perguntas ao tutor. Estas perguntas são limitadas por opções previstas na tela deste módulo, não sendo utilizada em nenhum momento, pelo menos nesta versão, linguagem natural na confecção destas perguntas.

As tabelas a seguir esboçam as telas que compõem alguns exemplos de exercícios do nível iniciante, servindo como modelo para implementação destes e de outros exercícios, no sistema.

Serão utilizados nas telas de exercícios os gráficos de um pentagrama e um teclado de cinco oitavas (quantidade de oitavas padrão nos teclados mais comuns do mercado), figuras 4.6 e 4.7, como espaço de resposta para o Aprendiz. No nível iniciante, e só neste nível, o teclado possui a marcação (legendas) com os nomes das notas musicais.

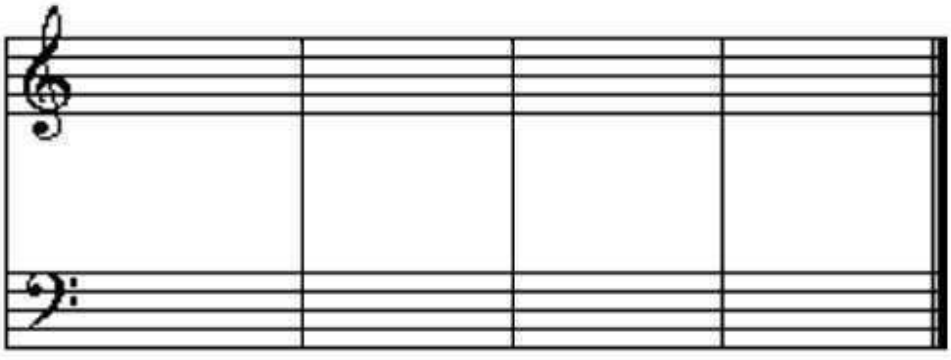


Figura 4.6 – "Pentagrama Limpo"

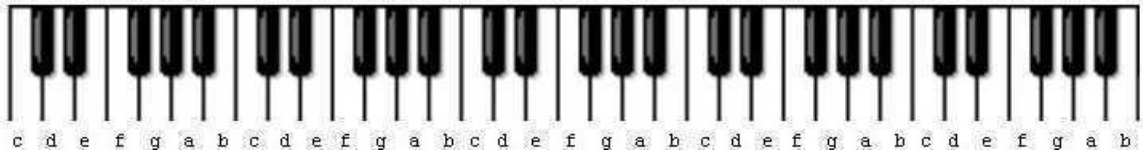


Figura 4.7 – "Teclado de Cinco Oitavas Legendado"

Tabela 4.6 – exercício para percepção do conceito de notas musicais (Nível Iniciante)

EXERCÍCIO 1	**TEORIA**	**EXEMPLOS**	**PERGUNTA**	**DICA**
Percepção de Notas Musicais	Os nomes das sete notas musicais naturais são: dó, ré, mi, fá, sol, lá e si. Um outro tipo de convenção, chamada de notação alfabética, se utiliza de letras para representar as notas musicais: dó (c), ré (d), mi (e), fá (f), sol (g), lá (a), si (b) Com as notas musicais é possível se criar melodias, acordes e estruturas musicais mais complexas.	(Nesta tela serão apresentados os gráficos de um pentagrama e um teclado, onde serão apresentados os eventos sonoros e gráficos de cada nota por um determinado período de tempo. O aluno pode pedir a repetição da apresentação.)	Para cada um dos cinco eventos sonoros apresentados, indique o seu valor clicando no pentagrama ou no teclado, ou ainda indicando o nome da nota e a sua alteração.	A freqüência sonora da nota é definida pelos atributos *nome*, *oitava* e *alteração* e *duração*. Quanto mais se ouvir as notas musicais repetidamente, mais acostumados os ouvidos estarão para perceber o valor de cada uma.

Tabela 4.7 – exercício figuras de ritmo (Nível Iniciante)

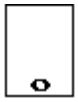
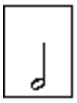
EXERCÍCIO 2	TEORIA – TELA 1	TEORIA – TELA 2	PERGUNTA	DICA
Identificação de Figuras e Duração das Notas Musicais	A duração das notas obedece uma proporcionalidade e uma relatividade entre os símbolos que representam o tempo. A duração e o ritmo dependem da figura de tempo utilizada. A distribuição espacial (sentido horizontal) destas figuras, sobre um pentagrama, indica o ordem cronológica dos acontecimentos. Já a altura das notas (freqüências) está relacionada com a disposição destas no sentido vertical. Quanto mais em cima, em relação ao pentagrama, mais aumenta a freqüência, o contrário também é verdadeiro: se a nota é colocada mais para baixo, diminui a freqüência.	 semibreve = 1  mínima = 2  semínima = 4  colcheia = 8  semicolcheia = 16  fusa = 32  semifusa = 64	Para cada nota apresentada no pentagrama, indique o nome da figura e o nome da nota, clicando nas opções disponíveis. (Serão apresentadas as notas no pentagrama e no teclado, o evento sonoro correspondente, além das opções de nomes para a escolha do Aprendiz)	

Tabela 4.8 – exercício intervalos (Nível Iniciante)

EXERCÍCIO 3	TEORIA – TELA 1	TEORIA – TELA 2	PERGUNTA	DICA
Intervalos	Intervalo é a distância entre duas notas. Quando as notas que compõem o intervalo se encontram no perímetro de uma oitava, tem-se um intervalo simples e, quando ultrapassa uma oitava, tem-se um intervalo composto.	**grau**: o grau do intervalo é medido pela distância que separa dois nomes de notas dentro da lista (c, d, e, f, g, a, b), repetindo-se em todas as oitavas do teclado.	Para cada par de notas apresentado, indique o nome e o grau do intervalo, clicando nas opções	Tabela 4.1: nomes dos intervalos

	Um intervalo possui como características: **natureza** (vertical ou horizontal): se o intervalo ocorrer em uma melodia, estará caracterizada a natureza horizontal do intervalo (melódico). Caso ocorra entre notas de melodias diferentes e simultâneas, estará caracterizado o intervalo vertical (harmônico). **direção**: de acordo com a posição da segunda nota, o intervalo pode ser ascendente (quando a segunda nota é mais aguda que a primeira), ou descendente (quando a segunda nota é mais grave que a primeira).	Os intervalos acontecem entre quaisquer nomes e a sua contagem deve incluir o primeiro nome, os nomes compreendidos entre este e o segundo nome. O resultado é o nome do grau. Número de Semitons: dependendo do grau do intervalo, o número de semitons irá estabelecer o restante do nome do intervalo (diminutos, menores, maiores, aumentados e justos).	disponíveis. (Serão apresentadas as notas no pentagrama e no teclado, além das opções de nome e grau para a escolha do Aprendiz. Serão apresentados três intervalos, com níveis de dificuldades crescentes, para cada interação)	e seus graus

Tabela 4.9 – exercício acordes maiores (Nível Iniciante)

EXERCÍCIO 4	**TEORIA – TELA 1**	**TEORIA – TELA 2**	**PERGUNTA**	**DICA**
Acordes Maiores	ACORDES são conjuntos de intervalos verticais (três ou mais notas), dispostos segundo as leis da Harmonia. Os acordes podem ser *consonantes* ou *dissonantes*, *maiores* ou *menores*, *tríades*, *tétrades*, de *sétima*, etc. Neste nível serão considerados	Similarmente às escalas, os acordes são "montados" segundo determinadas relações intervalares. Para resumir a construção de um acorde pode-se afirmar: • acordes maiores, menores, diminutos e aumentados são formados com três notas (tríades); • acordes de sétima são formados com quatro notas (tétrade); • todo acorde possui uma nota que lhe dá nome (inicialmente trabalha-se com acordes em posição de fundamental, ou seja, a nota que dá nome ao acorde é sempre a mais grave); • partindo da fundamental	Dadas as seguintes notas, que nota falta para caracterizar um acorde C maior ? (O acorde é sorteado pelo sistema e serão apresentadas as notas no pentagrama [fig 4.8] e no teclado [fig 4.9], para a	Tabela 4.5 - relação entre terças para formação de acordes

	apenas os acordes maiores, menores, diminutos, aumentados (estes constituídos por três notas) e de sétima (constituído por quatro notas).	(inferior), tem-se uma distância de 3º grau até a segunda nota (chamada de terça do acorde) e de 5º grau até a terceira nota (chamada de quinta do acorde). Por exemplo: partindo da nota C tem-se como 3º grau a nota E e como 5º grau a nota G (terças superpostas), formando o acorde de dó. Tratando-se de acordes de sétima, tem-se uma quarta nota posta a distância de 3º grau da quinta do acorde. Por exemplo, o acorde de dó com sétima terá a nota B acrescentada às anteriores.	escolha da nota faltante pelo Aprendiz)	

A tabela 4.9 mostra um esboço de exercício criado para acordes maiores. Abaixo temos uma seqüência de imagens que podem ser utilizadas em tutores em Harmonia para apresentar este exemplo de exercício. Neste caso, será perguntado ao aluno sobre o acorde maior de Dó (C).

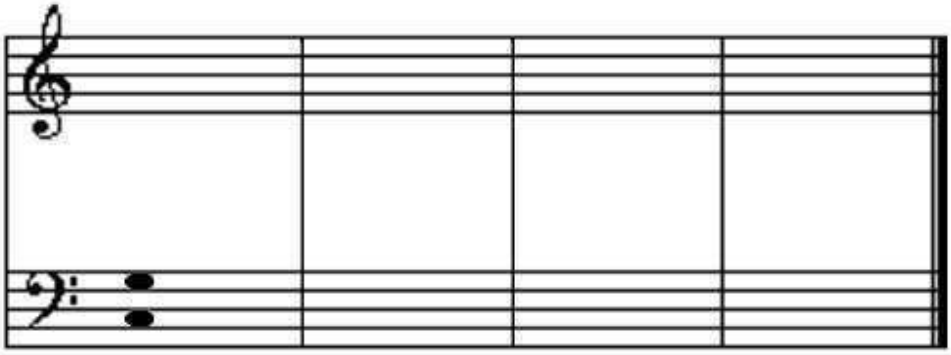


Figura 4.8 – Pentagrama Exercício Acordes Maiores, C

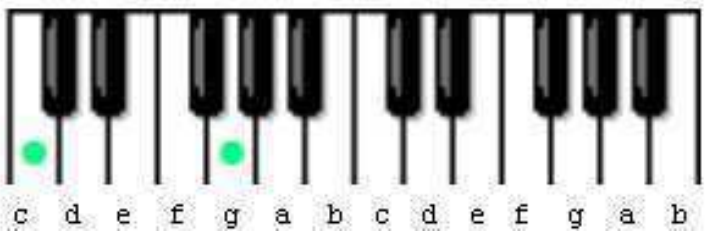


Figura 4.9 – Teclado Exercício Acordes Maiores, C

Na tela de avaliação deste último exercício, serão apresentadas as figuras 4.10 e 4.11, além da mensagem indicando se o aluno acertou ou onde errou.

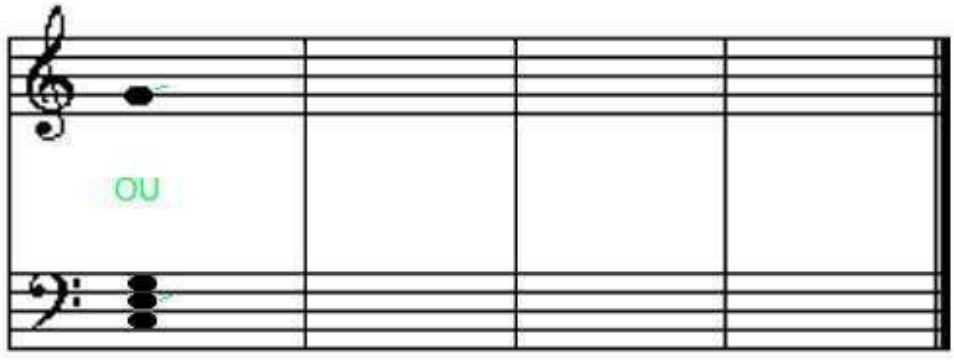


Figura 4.10 – Pentagrama Resposta Exercício Acordes Maiores, C

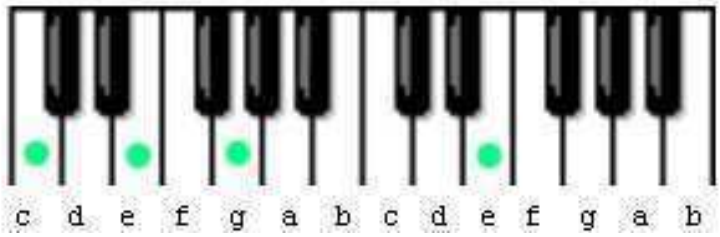


Figura 4.11 – Teclado Resposta Exercício Acordes Maiores, C

4.5. Conclusões

Apresentamos neste capítulo as soluções utilizadas pelo nosso tutor para a representação do conhecimento necessária para a base de conhecimento do sistema, onde utilizamos a representação descrita no capítulo 3, apresentamos a inclusão de uma interface

multimídia para uma melhor interação Aprendiz-Tutor, na intenção de alcançar melhores resultados pedagógicos.

Foram discutidos aspectos da Harmonia Musical utilizados no desenvolvimento do tutor, com ênfase nos exemplos de interação tutorial para sistemas neste domínio, com a apresentação de exercícios e exemplos incluídos no nosso sistema.

No próximo capítulo será apresentado o MHITS, seus aspectos de modelagem e implementação, concluindo com a apresentação do protótipo e sugestões para novas versões do sistema.

Capítulo 5
Um Sistema Tutor Inteligente em Harmonia Musical

5.1. Introdução

No capítulo anterior foi discutido o domínio da Harmonia Musical. Foram mostrados os conceitos básicos relativos a este domínio de conhecimento tais como notas musicais, intervalos, acordes, escalas e, principalmente, uma abordagem voltada ao desenvolvimento do tutoramento aplicado pelo sistema proposto neste trabalho. Foram esboçados exemplos de exercícios que podem ser utilizados em tutoriais sobre Harmonia ou Teoria Musical, e que foram incluídos e melhor modelados durante o desenvolvimento do nosso sistema.

Neste capítulo apresenta-se a descrição do MHITS – *Musical Harmony Intelligent Tutoring System*, um ambiente para o auxílio à aprendizagem da Harmonia Musical.

Visando um melhor entendimento do sistema, aborda-se aqui a sua descrição geral, seus requisitos básicos de dados, sua modelagem baseada em objetos e sua implementação baseada em agentes, discutidos e comentados individualmente. A apresentação do protótipo desenvolvido, suas telas e funcionamento, será feita em paralelo à apresentação de cada módulo respectivo.

5.2. Descrição Geral do MHITS

5.2.1. Motivação e Objetivos

O ensino de Harmonia nos moldes tradicionais, com aulas em sala, exercícios práticos acompanhados pelo professor, lousa e giz, é eficiente e comprovado pelo fato de termos bons músicos e excelentes compositores formados pelas grandes escolas de música da atualidade. Mas há ainda a falta de acompanhamento do aluno em exercícios práticos além dos definidos

em sala de aula e de mecanismos de tutoramento disponíveis para a aprendizagem extra-classe, na escola ou na casa do aluno. A idéia de se desenvolver um tutor em Harmonia Musical tem como motivação principal o desenvolvimento de um ambiente de apoio ao estudo da Harmonia pelo aluno, que possa ser usado em qualquer hora, em qualquer lugar, desde que tenha um computador para auxiliá-lo. A construção de sistemas deste tipo não visa a substituição de professores humanos mas sim um auxílio ou um complemento para a aprendizagem e fixação do conteúdo ministrado. Há ainda a possibilidade de se criar cursos onde o sistema e a Sociedade de Especialistas Humanos (SEH, capítulo 3) sejam consultados pelos aprendizes via Internet, acessando-os remotamente, aumentando a possibilidade de interação com mensagens via correio eletrônico e servidores de *chats* (salas virtuais de bate-papo) configurados especialmente para este fim.

A partir da representação voltada ao ensino de Harmonia Musical desenvolvida em [TEIX 97] e do modelo MATHEMA para sistemas tutores inteligentes [COST 97], tendo por outro lado a ausência de tutores multimídia em Harmonia Musical, desenvolveu-se a proposta do MHITS. O desenvolvimento de um sistema que utilizasse a multimídia na comunicação com o Aprendiz, com a aplicação de técnicas oferecidas pela IA para possibilitar a representação do conhecimento neste domínio e principalmente para a elaboração de respostas às questões do usuário e ainda que contemplasse características para a sua utilização em programas de educação à distância , foi o objetivo principal deste trabalho.

Como objetivos secundários temos o desenvolvimento de um mecanismo de comunicação entre o módulo em LPA-PROLOG++ [VASE 95] e os agentes desenvolvidos em JAVA [ECKE 97], [JEPS 97], [LEMA 96], [LIND 98], [ROBE 97], além de uma modelagem e estruturação física do sistema visando uma utilização remota via Internet/Web.

5.2.2. Sobre os Usuários e Especialistas do MHITS

O ambiente MHITS é direcionado para estudantes iniciantes em Harmonia Musical, construído para ser utilizado por estes alunos ou aprendizes nas escolas de música e em casa, dando continuidade aos ensinamentos e exercícios ministrados pelo professor em sala de aula. O MHITS também poderá ser usado durante as aulas, conforme a necessidade do professor ou responsável pelo curso. Estes cursos podem também ser ministrados à distância, via Internet,

onde o sistema estará instalado em um servidor na Internet e os alunos e participantes do curso poderão acessar o sistema utilizando um browser web compatível com JAVA e serviços para comunicação com o instrutor humano durante o tutoramento.

Os usuários mais comuns do MHITS são os alunos e os professores, além do gerente do sistema, estes dois últimos com poderes totais para alterações nos dados e configurações internas do sistema.

O direcionamento para alunos iniciantes não é obrigatório já que o sistema apresenta três níveis para os alunos: iniciante, médio e avançado. Mas, como usuário anônimo, via Internet, o usuário terá que se cadastrar obrigatoriamente como iniciante e terá que passar por todas as teorias e exercícios do sistema, todas as vezes que quiser utilizá-lo. Já o aluno cadastrado pelo professor para um curso on-line ou para uso nas aulas de exercício em laboratório ou em casa, pode ser cadastrado em qualquer dos três níveis, segundo o julgamento do professor ou responsável pelo curso.

O nível do aluno avança a medida que ele alcança sucesso nos exercícios e atinge o final do tutorial em cada nível. Um exemplo de continuidade e desafio no tutorial: para cada exercício que o aluno responder de forma errada, terá que responder dois do mesmo estilo, corretamente. Ele terá a sua disposição durante a resolução deste exercícios um atalho para dicas disponibilizado na forma de um botão de interface, além do Módulo de Perguntas, descrito mais a frente. O sistema prevê ainda a repetição dos exercícios, com parâmetros diferentes e exemplos diferentes para as mesmas perguntas, de forma que o aluno possa praticar mais e fixar melhor o assunto em questão.

Para utilizar o MHITS basta ter um computador a disposição, ser estudante ou interessado em Harmonia Musical, ser cadastrado pelo professor ou cadastrar-se como anônimo via Internet.

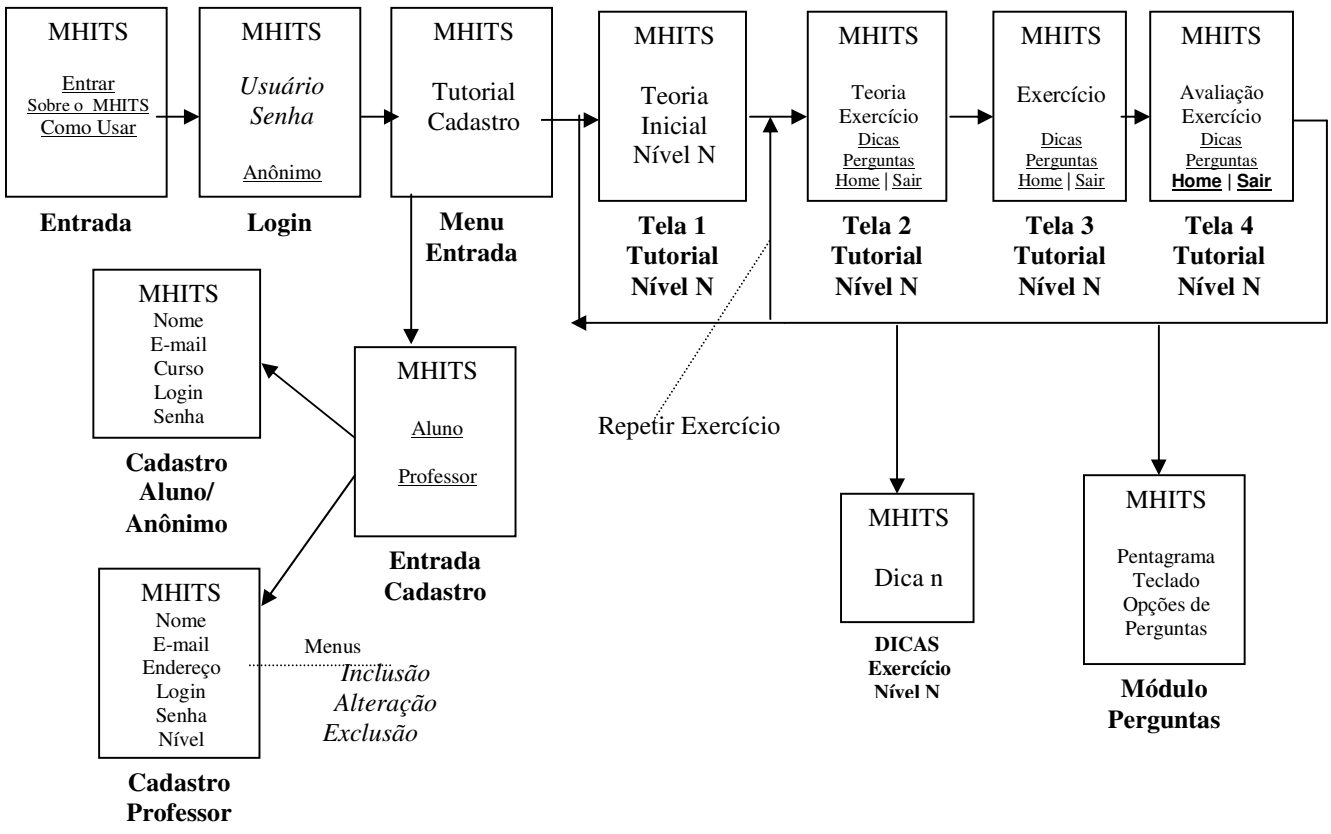
Logo na primeira tela do sistema, detalhada mais a frente, o usuário tem acesso a informações em hipertexto sobre o sistema e como usá-lo, facilitando a interação inicial e adaptação ao formato e estilo do tutorial adotado. Este formato do tutorial e os exercícios apresentados seguem o formato básico encontrado em diversos livros e apostilas de cursos de Harmonia e Teoria Musical, mas deve ser adaptado, em versões posteriores do sistema, para

cada currículo de cada instituição que desejar utilizá-lo.

5.3. Interações no MHITS

O esboço desenvolvido em [TEIX 97] atingiu os objetivos de representação e tem na comunicação textual no próprio ambiente Prolog o único modo de interação Aprendiz-Tutor. Este Tutor foi apresentado no capítulo 3 e serviu como base para o desenvolvimento do MHITS, onde foram agregadas características multimídias às interações Tutor-Aprendiz, incluindo um esquema de navegação hipermídia pelas opções do sistema, onde o usuário pode, a partir de uma imagem ou texto, visualizar outros recursos como textos, telas de interação, exemplos sonoros, além de outras imagens.

Neste tópico apresentaremos o ambiente MHITS através das suas possibilidades de interação, comentando sucintamente alguns módulos necessários a esta interação. O diagrama de interação MHITS aparece na figura 5.1, onde temos um esquema lógico dos caminhos que podem ser percorridos pelo Aprendiz durante sua comunicação com o sistema, além de visualizarmos todas as possibilidades de navegação pelos módulos do sistema.



Na primeira tela do sistema temos a tela de entrada, onde o usuário pode optar por entrar no sistema (botão "Entrar"), escolher entre saber mais sobre o ambiente (botão "Sobre o MHITS") ou saber como usar o sistema (botão "Como Usar"). Esta é uma tela básica inicial para um sistema hipermídia qualquer. Existe ainda a opção de fechar a janela do sistema antes mesmo de qualquer interação, através dos botões padrão de janelas baseadas no Windows ™, a saber, minimizar, maximizar e fechar janela. O *Agente da Interface* controla todas as informações que serão mostradas ao usuário, bem como comunica-se com os demais agentes, servindo como mediador entre o usuário e os mecanismos internos do ambiente. Este agente será melhor descrito mais adiante.

Ao clicar nos botões "Sobre o MHITS" e "Como Usar", um browser hipertexto é ativado, a fim de que o usuário obtenha, através da leitura e aprofundamento nos tópicos desejados, informações sobre o sistema e sobre como usá-lo obtendo melhores resultados.

Entrando no sistema (botão "Entrar"), o usuário necessitará informar o seu login e sua senha única de acesso cadastrada anteriormente pelo seu professor ou optar por navegar no sistema como anônimo, tendo, neste caso, a opção de se cadastrar como aluno iniciante temporário, para fins de avaliação do sistema. Ao informar o login e a senha ao sistema, o usuário irá para a terceira tela, onde poderá escolher entre ativar o *Módulo de Cadastro* (botão "Cadastro") ou entrar no tutorial propriamente dito (botão "Tutorial"). A opção de ativar o cadastro do sistema tem vários níveis de segurança, onde: *(i)* apenas o Gerente do sistema pode manipular TODAS as fichas do cadastro, inclusive a dos professores, *(ii)* apenas os professores podem cadastrar alunos com níveis diferenciados, exclui-los e modificá-los, *(iii)* os alunos podem acessar as suas próprias fichas e visualizar os outros inscritos por nome e login, mas não podem fazer mais nada além disso, *(iv)* os anônimos podem cadastrar-se como alunos iniciantes, para fins de avaliação do sistema e estatísticas internas de uso do mesmo.

A opção para acessar o tutorial está disponível praticamente para todo e qualquer usuário do sistema, seja cadastrado ou anônimo. Esta decisão de projeto proporciona um período de avaliação e cadastramento de interessados no sistema, bem como *feedbacks* destes usuários, com sugestões e reclamações, através da opção de e-mail para o gerente do sistema, apresentada no final de qualquer interação.

No *Módulo Tutorial*, descrito mais adiante, são apresentadas sempre interações compostas de quatro fases:

1. **Telas contendo teoria inicial sobre o conteúdo abordado no nível** – esta primeira fase contempla a exposição sobre os tópicos necessários ao Aprendiz para a aquisição de conhecimento relativos ao nível em que ele se encontra e no qual terá que trabalhar a parte prática com exercícios respondidos corretamente, afim de obter um avanço no seu nível. Esta teoria é apresentada apenas no início de cada nível;

2. **Telas contendo teoria sobre cada tópico da Harmonia Musical** – esta segunda fase engloba os tópicos teóricos dentro do nível em que o aluno se encontra e que servirão como base para o exercício que será apresentado na próxima fase. Este formato de teoria específica para cada tópico e respectivamente para cada estilo de exercício, é repetida no início da interação de cada exercício, quando o Aprendiz optar por continuar o tutorial;

3. **Tela com exercício sobre o tópico anterior** – apresenta um estilo de exercício correspondente ao conteúdo apresentado no item anterior, com um amplo espaço interativo para o Aprendiz, com espaço de resposta contendo tanto um pentagrama como um teclado de cinco oitavas, além de opções de botões ou questões multi-escolha, onde poderá marcar uma ou várias opções dependendo do exercício. O estilo de exercício será sempre o mesmo mas com dificuldade e tema (nota, tonalidade, tônica do acorde, etc) controlados pelo *Módulo Tutorial*, descrito mais a frente. O Aprendiz tem ainda as opções por dicas sobre o exercício, ativar o *Módulo de Perguntas*, voltar para a tela inicial do sistema ou sair do sistema desistindo da interação neste ponto.

4. **Tela com avaliação do exercício anterior** – comenta a resposta do aluno (identifica a resposta) e mostra as figuras da resposta correta caso o aluno tenha acertado a questão. Caso o aluno tenha errado, é apresentada uma dica para resolução do problema, uma opção para repetir o exercício e uma opção para retornar à apresentação da teoria sobre o exercício (fase 2 do exercício). Esta tela mantém ainda a opção para desistir da interação neste momento. Caso o aluno

tenha obtido sucesso na conclusão do nível, é apresentada ainda mais uma tela com uma retrospectiva resumida da sua interação neste nível que acabou de vencer, com opções para continuar no próximo nível ou parar.

Ao desistir da interação em qualquer momento do Tutorial, o Aprendiz poderá retornar a qualquer momento reiniciando aquele tópico onde parou a partir da apresentação da teoria específica daquele exercício. Isto é possível porque o sistema guarda toda a interação do aluno para a avaliação do professor, armazenando os dados pertinentes no final da última fase de cada tópico, quando o aluno opta por continuar ou repetir o exercício ou parar naquele ponto, ou ainda no final de cada nível, quando o aluno alcança o êxito no último exercício do nível. O Agente Tutor é responsável pelas mensagens de armazenamento dos dados no Modelo do Aluno.

Está prevista ainda durante o tutorial, a apresentação de dicas específicas para cada exercício, apresentadas em telas a parte ou em pop-ups (pequenas janelas de browser com conteúdo específico). Estas telas podem conter tabelas, diagramas, figuras, exemplos sonoros, texto e outros, que possam iluminar o caminho do Aprendiz para a solução do problema sem entregá-la facilmente, fazendo-o encontrá-la por si próprio.

Uma parte importante na interação e talvez um dos pontos mais importantes desta versão do MHITS, é o *Módulo de Perguntas*, através do qual o Aprendiz poderá construir perguntas a partir de um espaço apresentado em uma tela específica contendo a figura de um pentagrama, a figura de um teclado, opções de tonalidade e alterações de notas, e ainda perguntas comuns ao tutor como "Deseja perguntar: -Que nota é esta ? -Que acorde é este ? -Que escala é esta ?" e outras. Este módulo utilizar o Agente Comunicador para perguntar à base de conhecimento no formato utilizado para representação das classes comentadas no capítulo 3, aguardar a sua resposta e mostrá-la formatada pelo Agente de Interface.

Exemplo de interação com o Módulo de Perguntas:

1. Usuário clicou na segunda linha, de baixo para cima, no pentagrama de G (Sol) para iniciar uma pergunta;
2. O Módulo de Perguntas troca a parte da imagem da segunda linha por uma segunda linha com a nota correspondente;
3. O usuário escolhe " Deseja perguntar:" marcando "Que nota é esta ?";

4. O usuário clica em “Enviar pergunta”" e o Módulo de Perguntas chama o Ag. Comunicador que por sua vez passa a string de comando Prolog "nota([1,g,1,1])" para a Base de Conhecimento; no caso de acordes, intervalos ou escalas será passado o comando correspondente com uma lista de notas, para que o Prolog retorne a resposta, se existir;
5. A Base de Conhecimento responde, na sua linguagem, para o Agente Comunicador, que chama o Agente de Interface para mostrar a tela de resposta e um link para dicas sobre aquela estrutura, ou uma mensagem de erro, no caso da resposta não fazer parte do escopo do conhecimento do sistema ou da estrutura solicitada pelo aluno seja impossível na Harmonia Musical.

O Módulo de Perguntas é uma interface direta para a Base de Conhecimento do sistema e permite explorar as características de inteligência agregadas com a representação do conhecimento em dimensões e lateralidades, onde cada classe que faz parte desta base busque o conhecimento necessário à solicitação do Aprendiz através de interações internas e resoluções em níveis diferentes do conhecimento, até que a resposta correta seja encontrada. Este módulo é descrito adiante e é uma das principais contribuições para a interface do ambiente.

O Modelo do Aluno prevê ainda a impressão de relatórios onde o professor poderá optar por acompanhar o desempenho de determinado aluno durante as interações em determinado nível, o desempenho de uma turma em determinado nível, acompanhar todas as interações de um aluno até o presente momento, ou todos os alunos, de acordo com o seu interesse. Está disponível ainda a opção para impressão das fichas dos alunos que atenderem a uma determinada faixa de interesse como: alunos cadastrados nos últimos n dias, alunos com cadastro via login anônimo, entre outras.

5.4. Requisitos Básicos do MHITS

5.4.1. Requisitos de Dados

Antes de iniciar a aprendizagem e prática com a utilização do MHITS, os usuários precisam ser cadastrados, a fim de que o sistema seja alimentado com uma situação inicial em relação aos Aprendizes e Professores (Especialistas Humanos) que utilizarão o sistema em determinado curso de Harmonia Musical. Estes dados são inicialmente informados ao sistema

através do Módulo de Cadastro, formatados em tabelas em texto plano, e armazenados no servidor do sistema. Abaixo temos exemplos de algumas tabelas de dados utilizadas no armazenamento desses dados:

TABELA ALUNO		
Campo	***Tamanho máximo***	***Tipo***
matricula	10	int
nome	30	string
idade	2	int
curso	30	string
e-mail	30	string
nivel	(**i**)niciante, (**m**)edio OU (**a**)vancado	char
login	14	string
passwd	8	string
pontuacao	6	int

Tabela 5.1: dados do aluno

A tabela do aluno guarda dados para o acompanhamento do avanço dos alunos, seu desempenho, sua pontuação desde que foi cadastrado e o seu nível, atributo mais importante para a definição das interações durante o tutorial. Este campo só pode ser alterado pelo professor responsável ou pelo sistema, após o final de cada conjunto de exercícios acertados dentro de um determinado nível.

O usuário que entra no sistema como anônimo pode se cadastrar para o tutorial mas só poderá fazê-lo como *aluno iniciante*. Esta ficha tem um tempo de avaliação de 30 dias, onde o anônimo poderá participar novamente de interações tutoriais, se assim o desejar. A idéia é obter uma estatística de uso do sistema, obtida também com usuários acessando remotamente, além da possibilidade de solicitar, em algum momento da interação ou quando o usuário desejar sair do sistema, um feedback rápido respondido diretamente via página web, em um formulário de entrada de dados comum aos web sites de hoje.

A tabela 5.2 mostra o formato de dados para o armazenamento de informações sobre os professores e especialistas humanos.

TABELA PROFESSOR		
Campo	***Tamanho máximo***	***Tipo***
codigo	10	int
nome	30	string
curso	30	string
e-mail	30	string
login	14	string
passwd	8	string

Tabela 5.2: dados do professor

Durante as interações o sistema guarda as informações sobre cada exercício e cada prática do aluno, relacionando os pontos conseguidos naquele exercício e a resposta escolhida naquele momento. Esta talvez seja a maior tabela do sistema, que servirá para obter, a qualquer momento, um histórico do aluno para uma avaliação feita pelo sistema (por exemplo ao final de um nível) ou impressão de um relatório para acompanhamento do professor. O modelo de dados relativo às interações está descrito na tabela 5.3.

TABELA INTERAÇÃO		
Campo	***Tamanho máximo***	***Tipo***
numeroInteracao	10	int
numeroExercicio	2	int
nivel	(**i**)niciante | (**m**)édio | (**a**)vançado	char
acertou	T|F	char
pontos	3	int

Tabela 5.3: dados das interações

A tabela Relacionamento Interação/Aluno guarda o relacionamento entre cada interação e cada aluno que participou da mesma. Estes dados estão representados na tabela 5.4. É preciso saber também quem é o responsável sobre cada aluno matriculado, já que podemos ter cursos com mais de um professor ou instrutor. A tabela 5.5 demonstra este relacionamento.

RELACIONAMENTO INTERAÇÃO/ALUNO	
numeroInteracao	matricula

Tabela 5.4: dados de cada interação relacionados a cada aluno

RELACIONAMENTO ALUNO /RESPONSAVEL	
matricula	codigo

Tabela 5.5: dados de cada aluno relacionados ao seu professor responsável

Os exercícios no MHITS são carregados durante a instalação mas há ainda a idéia de se implementar um sistema administrativo de exercícios, para alteração, exclusão e inserção dos mesmos pelos Especialistas Humanos. Com este intuito podem ser criadas ainda outras tabelas de dados mais complexas como uma tabela de exercícios, uma tabela de dicas e o relacionamento entre elas. A complexidade da primeira está na variedade de tipos de exercícios, as figuras mostradas para interação e resposta do aluno e a estrutura interna que captará esta interatividade (cliques do usuário), as perguntas, as possíveis tônicas, tipos de acordes e de escalas, tipos de intervalos e tonalidades, que podem ser utilizados na montagem do exercício pelo sistema. Muitos desses são campos difíceis de se dimensionar e de se definir o tipo de dados para armazenamento sem a possibilidade de comprometimento com a liberdade na criação e inserção de novos exercícios.

A escolha da modelagem dos usuários do sistema (Aprendiz e Especialista Humano) em uma base de dados é uma alternativa à modelagem lógica do aluno, comum em STIs tradicionais e que permite uma melhor avaliação *do* mesmo além de uma melhor adaptação do sistema *ao* mesmo, deixando esta tarefa não tão simples como um desafio para trabalhos futuros ou novas versões do MHITS.

5.4.2. Requisitos de Software e Hardware

O ambiente do MHITS requer os seguintes recursos para funcionamento:

1. *Servidor Prolog* (ligado a Internet e com recursos de memória e processamento apropriados), com sistema operacional Windows 9x, disponível para instalação do LPA-Prolog++ e que representará a base de conhecimento do sistema;

2. *Servidor HTTPD* (para transferência de hipertexto), também chamado de Web Server, que controlará as transações entre as aplicações clientes dos Aprendizes e os componentes do MHITS, suas bases de recursos externos e solicitações ao cadastro de usuários; é neste servidor que estarão instalados os componentes JAVA do MHITS, inclusive o ambiente JDK 1.1.8, *JAVA Development Kit* da *Sun Microsystems*, que foi utilizado na implementação do sistema e que possui os componentes e bibliotecas necessárias ao pleno funcionamento do sistema;

3. *Servidor mSQL,* para a instalação da base de dados e acesso pelos Agentes do MHITS;

A figura abaixo mostra o esquema para o servidor MHITS:

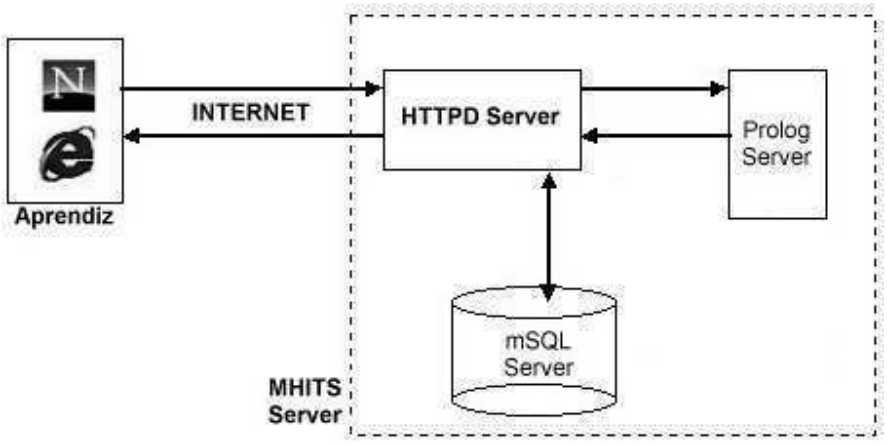


Fig 5.2 – Modelo Servidor MHITS

Os três servidores estarão instalados em uma única máquina, com sistema operacional Windows™ configurado para *enhanced mode* (suporte para aplicações 32 bits), e os seguintes

recursos mínimos de hardware [VASE 95]: processador Pentium 200 MMX, 64 MB de RAM, conexão Internet dedicada (para o caso de treinamentos remotos).

Para o uso em programas de treinamento a distância, está prevista ainda a instalação de um servidor de chats (salas de bate-papos virtuais) para interação direta entre os alunos do curso e o(s) instrutor(es). Estes servidores são facilmente encontrados na rede, com código aberto e livre para utilização, requerendo do administrador do sistema o conhecimento da linguagem de implementação dos componentes deste recurso (geralmente implementados em Perl ou JAVA) para adequação e instalação em um servidor ligado ao sistema.

5.5. Modelagem do Sistema

O sistema proposto aqui teve como modelo inicial um modelo simplificado do MATHEMA, discutido no capítulo 3, composto por uma Sociedade de Tutores, que procura responder às questões do Aprendiz através de uma consulta interna entre os seus vários componentes distribuídos em vários níveis de conhecimento, uma Sociedade de Especialistas Humanos, que alimentam o sistema com dicas, exercícios e outras informações, a Interface Adaptativa Multimídia, que se comunica com o usuário promovendo sua interação com o sistema e o Aprendiz, a peça principal de todo o sistema, o indivíduo para o qual pretende-se transmitir conhecimento. O Agente Comunicador promove a troca de informações entre a Interface e a Sociedade de Agentes Tutores.

O modelo inicial do MHITS está apresentado na figura 5.3, uma primeira visão dos objetos do sistema.

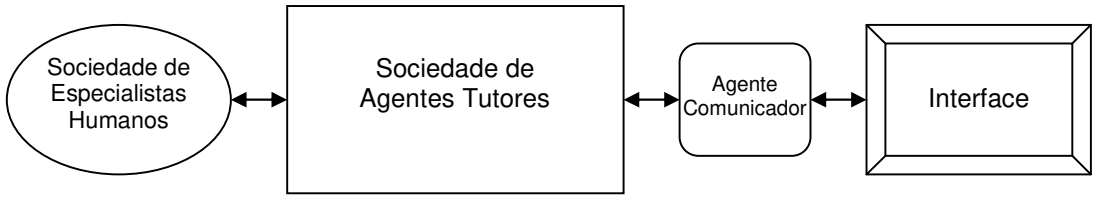


Fig 5.3: Modelo inicial do MHITS

A partir deste modelo inicial podemos, com algumas explosões, conseguir um nível de detalhamento consistente, adequado à implementação do sistema. O MHITS foi implementado a partir do modelo de objetos apresentado abaixo.

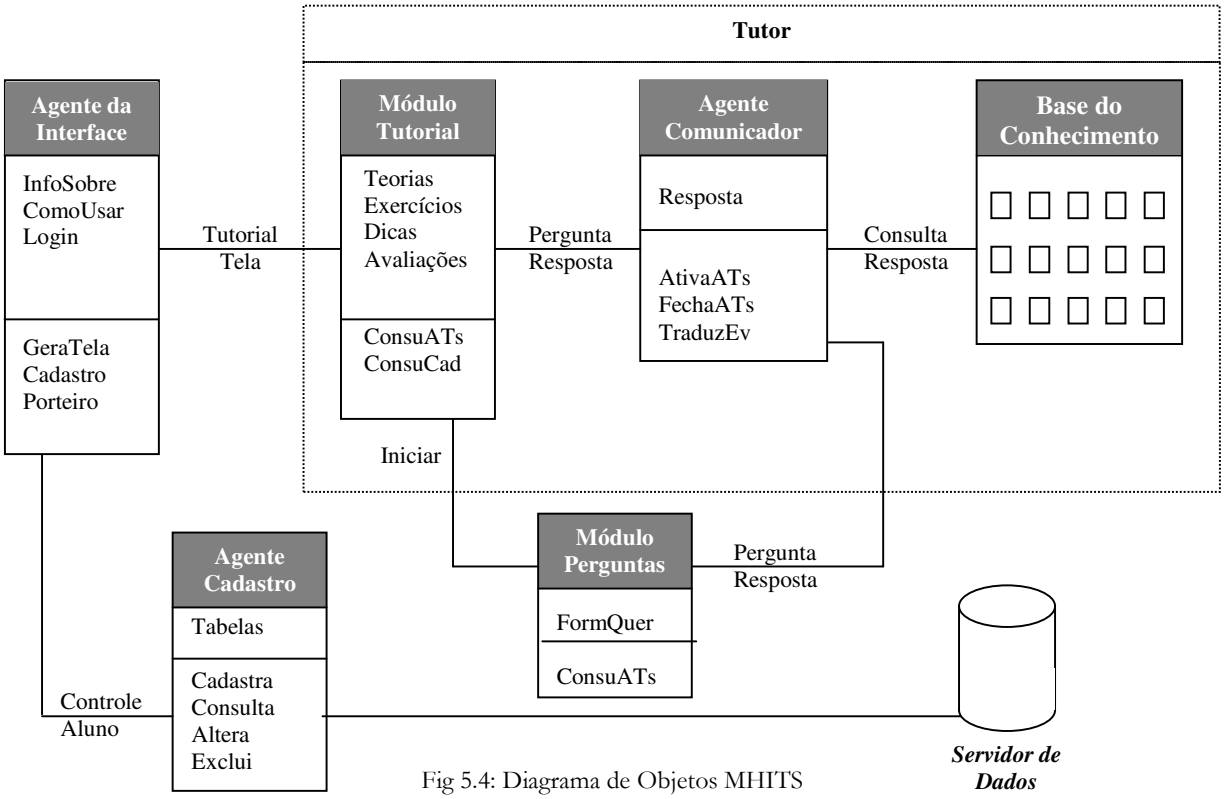


Fig 5.4: Diagrama de Objetos MHITS

Os componentes deste modelo estão descritos a seguir.

5.5.1. Agente da Interface

É o componente que permite a comunicação do Aprendiz com o MHITS. Implementado na forma de um applet JAVA, este agente controla o que será mostrado nos momentos iniciais da navegação pelo sistema, disponibiliza caminhos para a navegação hipermídia, inclusive por informações hipertexto sobre o sistema e como usá-lo. Controla a entrada (login) do usuário, identificando o seu nível de acesso às bases de dados dos cadastros. Possui elementos como janelas, botões, campos de entrada de dados, e utiliza arquivos armazenados na base Recursos Externos para apresentação de conteúdo multimídia para o usuário.

Através da Interface são disponibilizados objetos e áreas de interação, a partir dos quais são capturados dados, respostas, pedidos de ajuda e outros eventos de interação com o Aprendiz. Pode ser exibido através de uma janela típica de sistemas operacionais gráficos, ativadas por aplicações como o *appletviewer* (visualizador de applets) da Sun Microsystems, ou em um *browser web*, permitindo sua visualização local ou remotamente, da mesma maneira.

Abaixo são mostradas algumas telas deste agente para a visualização da sua organização e layout gráfico inicial. Devemos observar que todos os componentes gráficos utilizados pelo Agente Interface podem ser substituídos de forma a adicionar a identidade visual de cada instituição que o adotar, mantendo suas características de diagramação, nome e autoria do sistema.

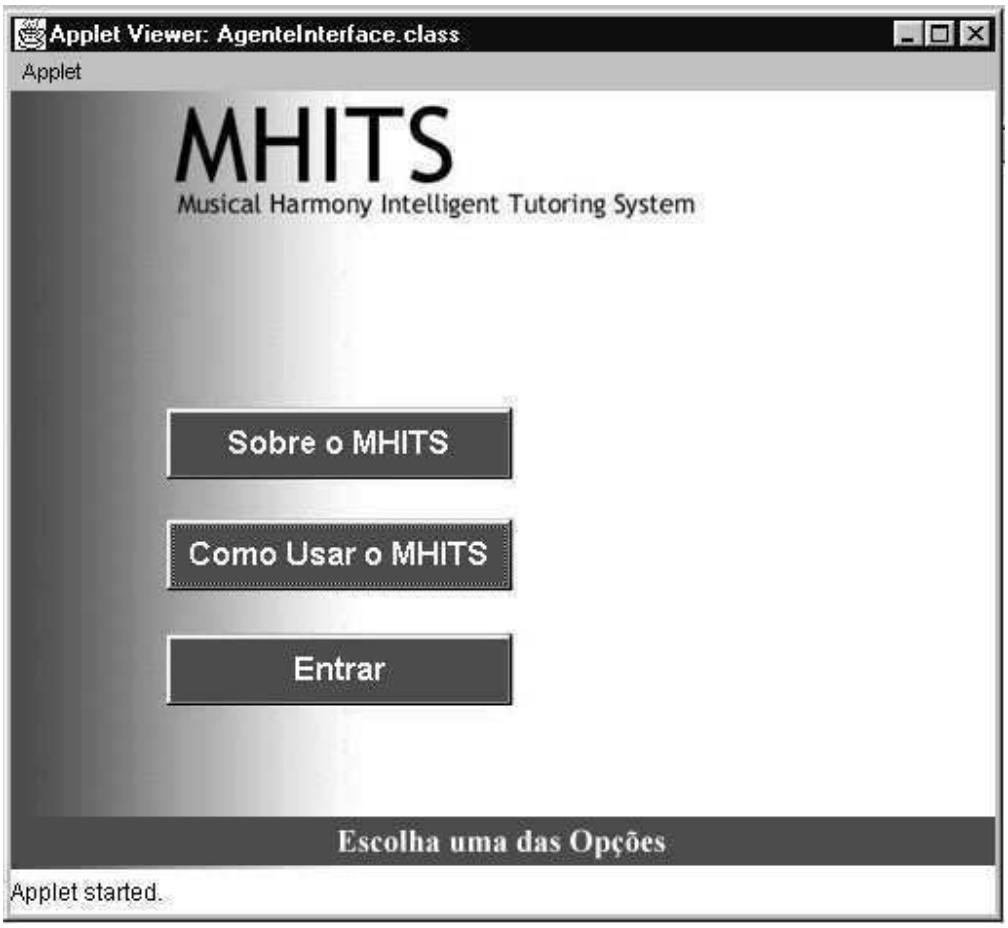


Fig 5.5: Tela de entrada do MHITS controlada pelo Agente da Interface

Fig 5.6: Tela de Login do MHITS controlada pelo Agente da Interface

O Agente Interface ativa o Módulo Tutorial passando os dados de entrada do Aprendiz para as interações tutoriais iniciais. Após a primeira interação o Módulo Tutorial vai interagir diretamente com a base dos cadastros para a adequação do exercício ao nível do aluno, bem como para modificar este nível, no momento oportuno. Na próxima figura é apresentada a terceira tela do Agente da Interface, com a opção para ativação do Módulo Tutorial.

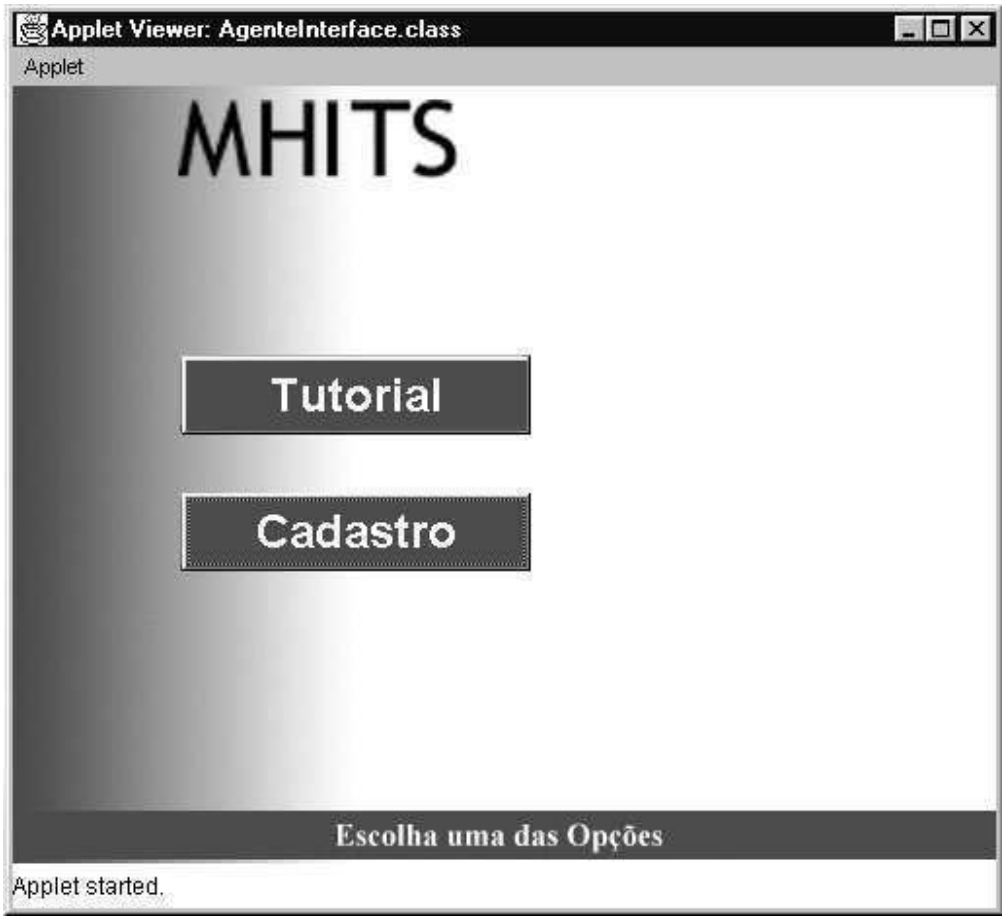


Fig 5.7: Tela de acesso ao Cadastro do MHITS e ativação do Módulo Tutorial

5.5.2. Módulo Tutorial

Este é um dos principais componentes do sistema. O Módulo Tutorial controla a forma como deve ser mostrado o material de tutoramento, gera as telas para cada sessão de aprendizado de acordo com o aprimoramento e nível do aluno, apresenta as perguntas que fazem parte do tutoramento através de várias mídias como gráficos, sons e texto, comunica-se com a Base de Conhecimento via Agente Comunicador, enviando as respostas do aluno e recebendo a resposta correta desta para apresentar ao aluno, além de gerar a configuração da próxima tela a partir desta avaliação.

Ao gerar as telas, este módulo consulta o estado do aluno, comunicando-se com o Agente Cadastro, para verificar o seu nível (iniciante, médio ou avançado). Quando o Aprendiz completa algum nível, sendo aprovado para o próximo nível, o Módulo Tutorial

avisa ao Agente Cadastro que este deve ser atualizado.

Durante a apresentação das teorias, exercícios, dicas e avaliações, o Aprendiz pode ativar, a partir de qualquer tela controlada por este agente, o Módulo de Perguntas, descrito mais a frente. Este módulo é uma ferramenta inédita neste tipo de sistema que serve para auxiliar na resolução dos problemas, aumentando a interação com o conteúdo apresentado e fazendo crescer a possibilidade de aquisição do conhecimento por parte do Aprendiz.

5.5.3. Agente Comunicador

Implementado em JAVA na forma de um *daemon*, efetua a comunicação entre o Módulo Tutorial e a base de conhecimentos via portas TCP/IP (protocolo de transferência de arquivos Internet). Este agente segue, de forma resumida, os seguintes passos: *(i)* recebe os eventos do aluno repassados pelo Módulo Tutorial ou as perguntas geradas pelo Módulo de Perguntas e os "traduz" para a linguagem dos ATs, *(ii)* pergunta aos ATs qual a resposta correta, *(iii)* recebe a resposta dos ATs e a disponibiliza para o agente que a solicitou.

O uso de portas TCP/IP garante o uso remoto por vários clientes (Agentes funcionando remotamente em browsers dos usuários) ao mesmo tempo, permitindo o uso do sistema em programas de treinamento ou cursos à distância.

A figura 5.8 esquematiza o modelo de comunicação entre os componentes do sistema que acessam a base de conhecimento através do Agente Comunicador. É usada como padrão a porta TCP/IP 4000, além dos diretórios *Queries* (solicitações) e *Answers* (respostas) para depósito de arquivos temporários necessários para comunicação.

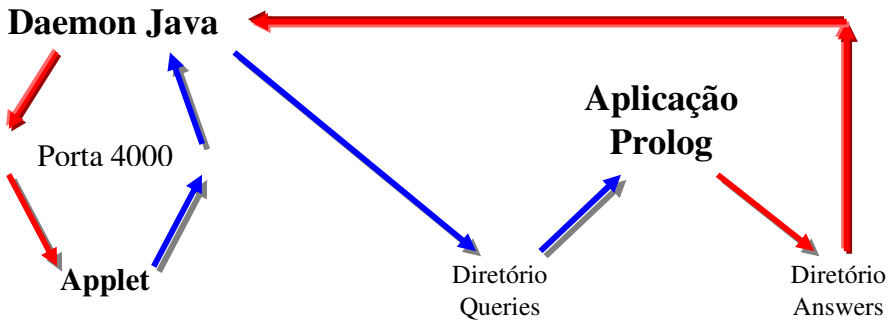


Fig 5.8: Esquema de Funcionamento do Agente Comunicador MHITS(*Daemon JAVA*)

5.5.4. Base do Conhecimento do MHITS

A base do conhecimento do MHITS, especificada e implementada em LPA-Prolog++ por Teixeira [TEIX 97], modificada e adaptada neste trabalho para interação com o Agente Comunicador, é responsável por informar a resposta correta às questões apresentadas ao aluno pelo Módulo Tutorial. Esta base recebe inferências do MHITS através do Agente Comunicador.

É solicitada quando o aluno responde uma questão e o sistema precisa da resposta correta para avaliar o que o aluno respondeu. Também é acessada quando o Aprendiz ativa o Módulo de Perguntas através do Módulo Tutorial, a fim de solucionar algumas dúvidas ou estudar um pouco mais sobre a teoria apresentada naquele momento pelo MHITS.

Seu funcionamento interno obedece as regras de comunicação entre agentes (representados aqui pelas classes Prolog++) nos moldes do MATHEMA [COST 97], onde cada componente é responsável por um aspecto ou subdomínio do conhecimento. Esta interação entre eles se dá na forma descrita no capítulo 3, onde cada agente procura compor a resposta para a inferência através da consulta a outros agentes detentores de um conhecimento mais específico, necessário para a solução da questão. A seção 3.3 deve ser consultada para maiores detalhes.

5.5.5. Agente Cadastro

Implementado em uma classe JAVA, controla as informações sobre o aluno, seu nível de aprendizagem e sobre os professores usuários do sistema. Possui um sistema de cadastro e consulta às informações que é modificado apenas pela Sociedade de Especialistas Humanos (responsáveis pelos alunos e administrador do sistema). É consultado pelo Módulo Tutorial para auxiliar na geração das telas de cada sessão a partir do nível do aluno. As figuras abaixo ilustra a tela deste módulo e as funcionalidades disponíveis.

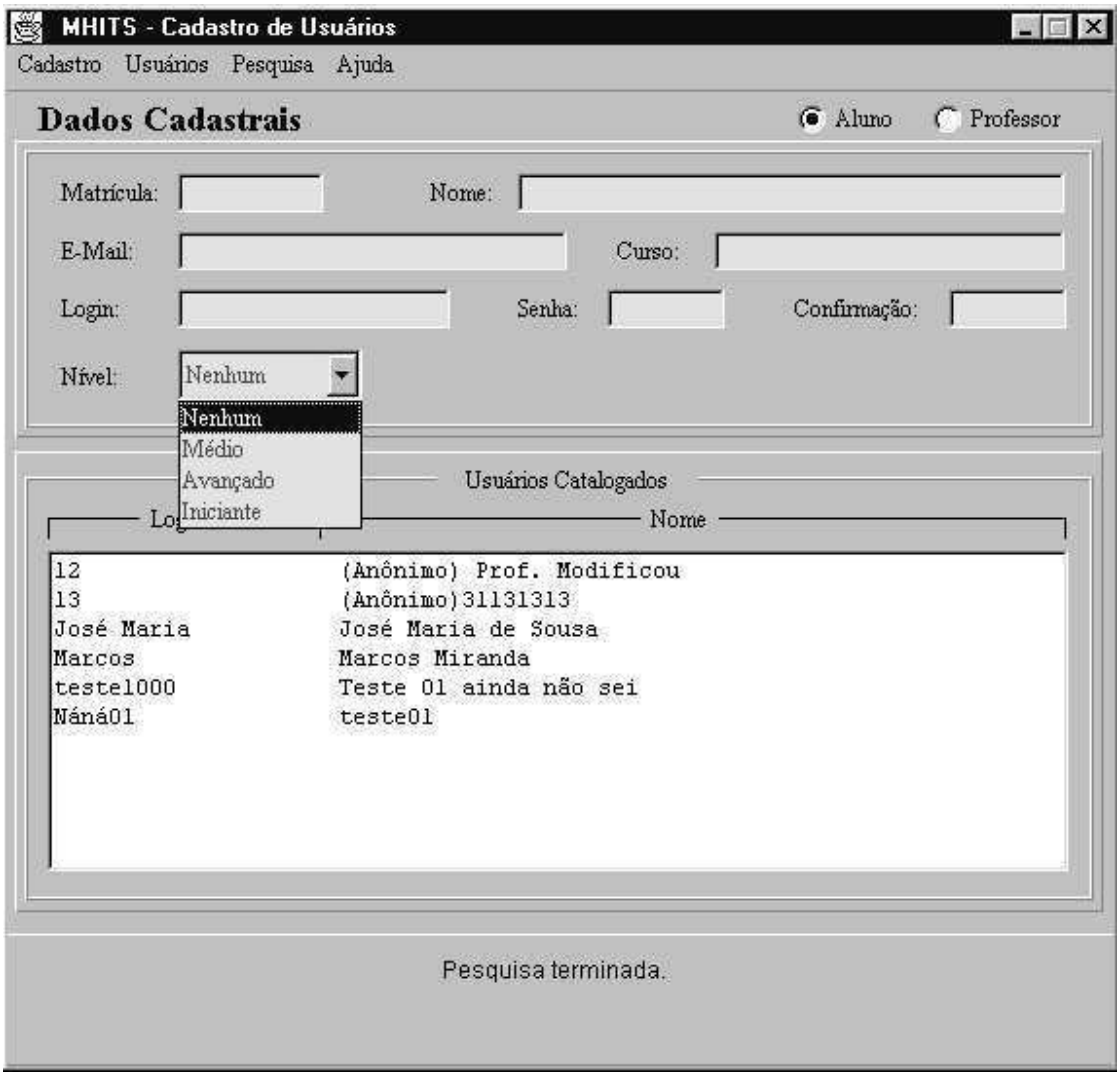


Fig 5.9: O Agente Cadastro do MHITS na inclusão de Aluno e escolha do nível inicial

Fig 5.10: O Agente Cadastro do MHITS na inclusão de novo usuário e escolha do tipo

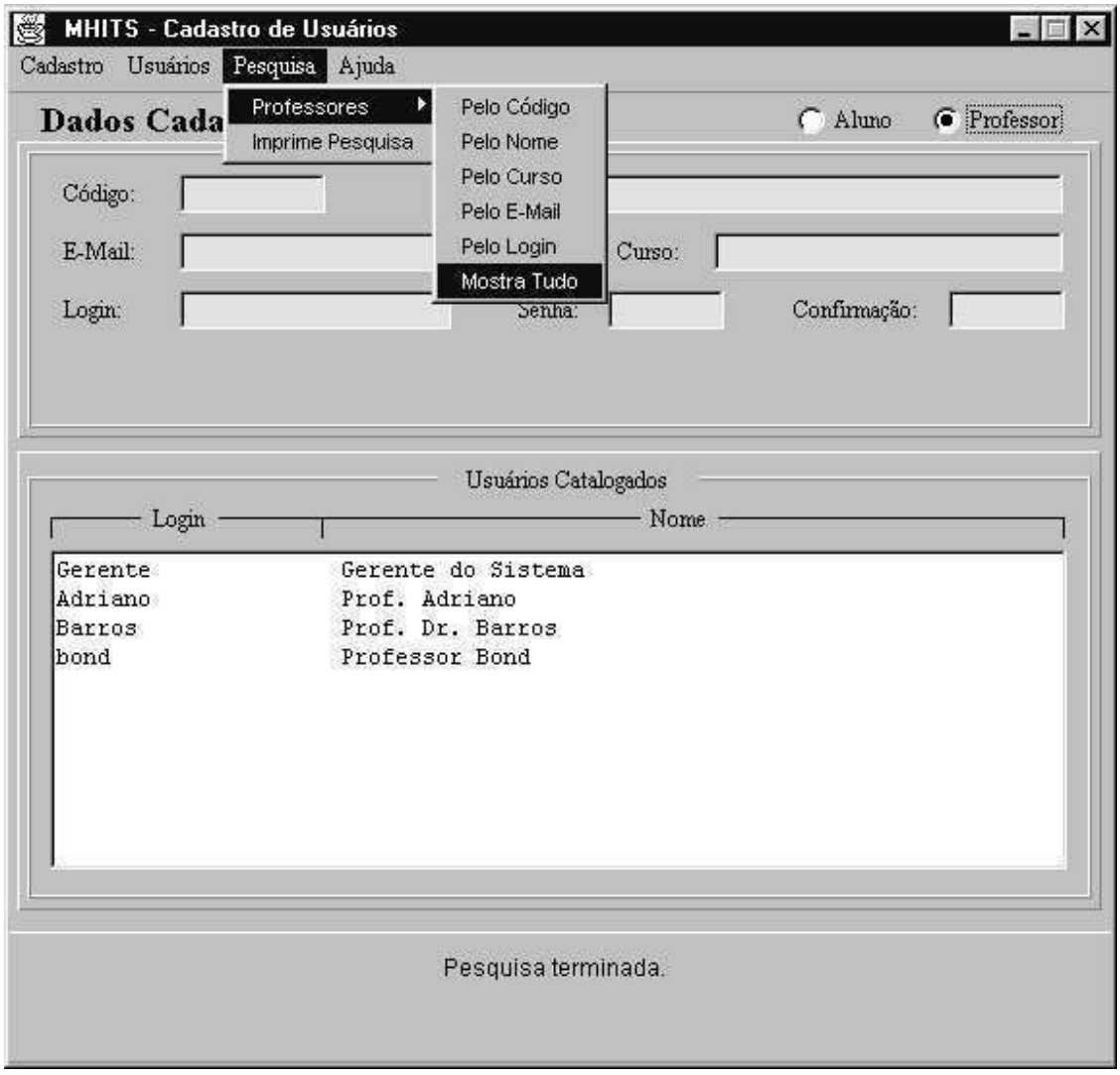


Fig 5.11: O Agente Cadastro do MHITS na pesquisa de usuários Professores

Este agente utiliza classes mSQL (*http://www.hughes.com.au/product/msql*) para manipulação da base de dados. Através desse agente, é possível imprimir relatórios sobre o desempenho dos alunos para acompanhamento.

5.5.6. Módulo Perguntas

Este módulo é uma interface multimídia que acessa a base de conhecimento através do Agente Comunicador para responder a questões do usuário formatadas segundo os objetos disponíveis na tela. Através dele é possível perguntar sobre determinada nota ou se um

determinado conjunto de notas compõem um acorde ou escala, ou outras perguntas do tipo, limitadas pelo formulário disponível para as perguntas. A figura abaixo mostra um esboço da tela deste módulo.

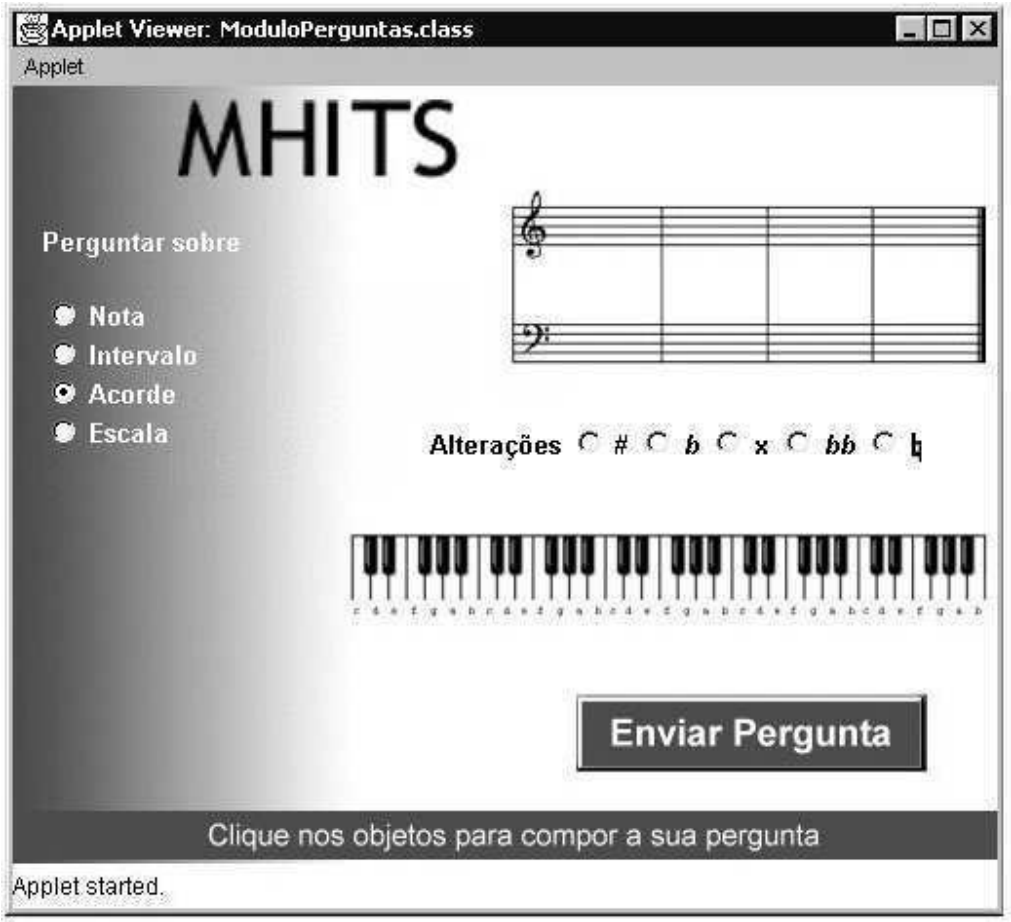


Fig 5.12: Esboço do Módulo de Perguntas MHITS

O MHITS utiliza ainda uma base de recursos externos que possui os arquivos necessários à formatação da Interface para interação com o usuário: imagens nos formatos GIF, JPG e PNG, arquivos HTML contendo tutoriais além de animações, arquivos de som MIDI.

O Agente Cadastro utiliza um servidor de dados para maior eficiência no acesso às informações sobre os usuários do sistema, principalmente o nível do aluno, consultado em tempo de execução pelo Módulo Tutorial.

Os componentes Módulo Tutorial, Agente Comunicador e Base do Conhecimento compõem a parte principal do sistema que é na verdade o próprio tutor. Estes componentes representam a Sociedade de Agentes Tutores Inteligentes (ATs), que faz parte do modelo MATHEMA, modelada e implementada neste trabalho de forma simplificada, rebuscando os modelos tradicionais de STIs compostos por Interface, Modelo do Aluno, Base de Conhecimento e Modelo do Tutor.

5.6. Aspectos Relativos à Implementação

5.6.1. Sobre as linguagens de desenvolvimento

A linguagem JAVA foi escolhida pela necessidade inerente de publicarmos recursos na Internet, utilizarmos recursos interativos multimídia, pela escolha da metodologia orientada a objetos na representação e especificação do sistema, por possuir recursos que nos permitiram a comunicação com a base de conhecimento Prolog. JAVA é uma linguagem moderna, em ascensão, voltada para Internet pois possui recursos prontos para programação distribuída, e seus programas podem ser utilizados em diferentes plataformas e diferentes sistemas operacionais. O LPA-Prolog possui recursos para a criação de interfaces, mas apenas dentro do ambiente Windows, sem a possibilidade de publicação via Internet, além das poucas opções para a utilização de recursos multimídia avançados como animações e vídeos. Com JAVA temos todos os recursos disponíveis para a web, em plena evolução e criação de novas tecnologias que podem ser adaptadas para uso no MHITS.

Os Applets JAVA são programas interativos e dinâmicos que funcionam dentro de uma página Web através de um browser compatível com JAVA, como o Netscape ou o HotJava. São transferidos via Internet para a máquina do usuário antes de serem executados. Os programas JAVA também podem ser implementados na forma de aplicativos mais gerais, que não necessitam de um browser para rodar. Segundo a *Sun Mycrosystems* (www.sun.com), sua criadora, JAVA pode ser definida como uma linguagem simples, orientada a objetos, fortemente tipada, compilada, independente de plataforma, com coleta de lixo, extensível, estruturada, robusta e segura. Os programas JAVA não têm acesso direto à memória e deixam o controle a cargo do sistema operacional, por isso, não causam core-dumps ou GPFs (*General*

Protection Fault). Um programa compilado em JAVA pode ser verificado antes de ser executado. A verificação dos *byte-codes*, formadores dos códigos compilados JAVA, é realizada nos browsers Web que suportam JAVA para garantir que os applets não estejam violando as restrições da linguagem e não possam provocar danos no computador do usuário.

Para desenvolver e fazer funcionar os seus programas é necessário um compilador JAVA além de outras ferramentas que o auxiliarão na tarefa de programador, como um visualizador de applets e um debugger. Estas ferramentas estão disponíveis no JDK – JAVA Development Kit, distribuído gratuitamente pela Sun Mycrosystems, ou em outros ambientes de desenvolvimento de aplicações e applets JAVA como o Symantec Café ou o JAVA Workshop.

O ambiente utilizado para o desenvolvimento do MHITS foi o JDK 1.1.8, disponível em http://www.java.sun.com/ . Esta versão foi escolhida por possuir todos os recursos e classes necessários à implementação do sistema.

Para a base de dados foi utilizado o Mini SQL ou mSQL, pacote gratuito desenvolvido por Joshua Dinerstein, disponível em http://www.hughes.com.au/product/msql . A versão utilizada foi a mSQL 2.0.4.1 for Win32. Este SQL foi utilizado por suprir todas as necessidades do sistema a nível de base de dados, por comunicar-se muito bem com aplicativos e applets JAVA, além de possuir um formato de implementação muito simples, com a introdução dos comandos SQL diretamente no código JAVA.

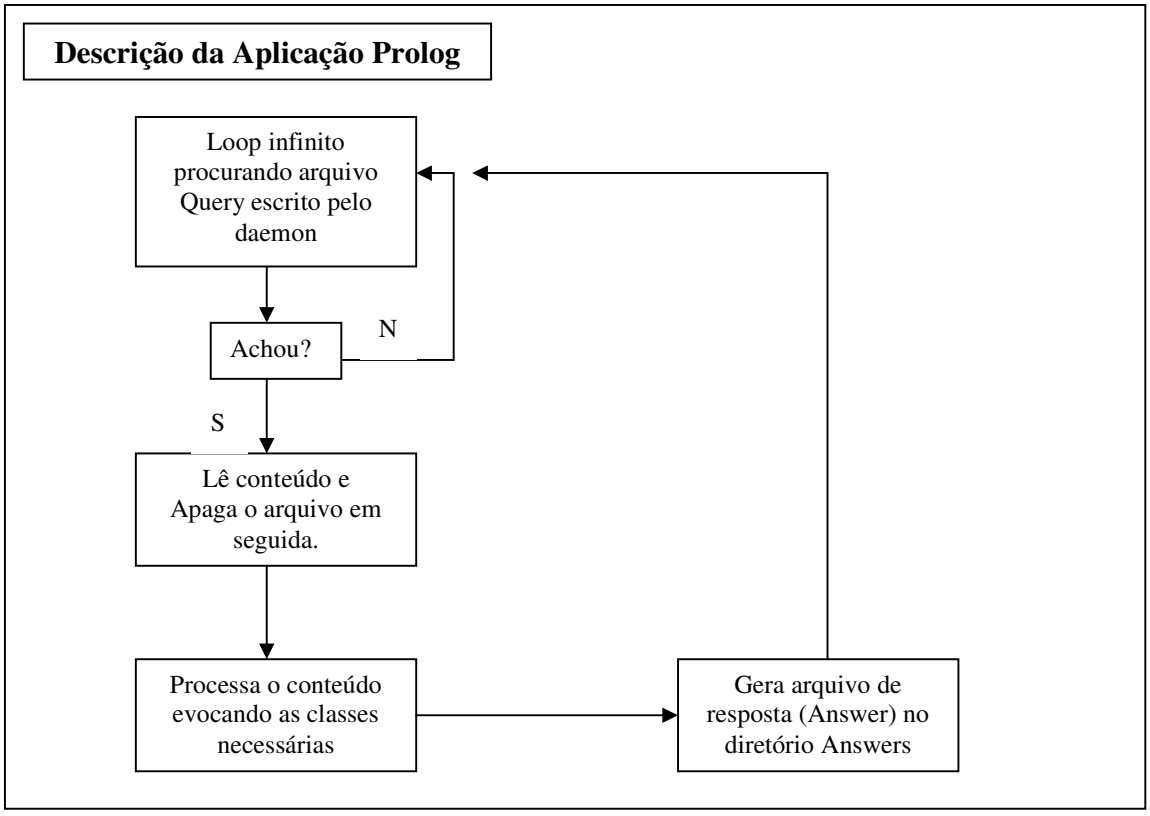
5.6.2. Considerações sobre a implementação dos módulos

Para esta versão, os exercícios, teorias e estruturas internas de controle da interação com o Aprendiz estão implementadas diretamente no código do Módulo Tutorial. Para uma próxima versão, pode-se prever a modificação deste módulo de forma que acesse estas informações a partir de uma base de dados (ou base de conhecimento), necessitando com isso da implementação paralela de um módulo administrativo para criação e alteração desta base.

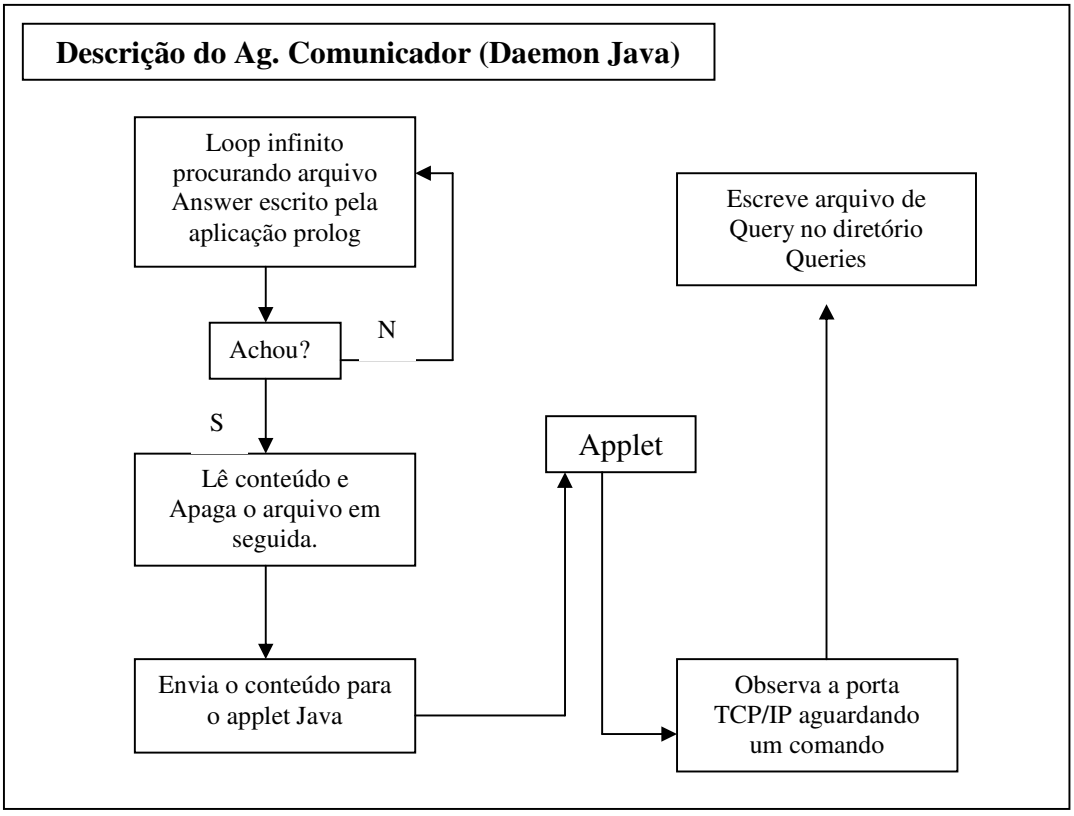
O Modelo do Aluno, muito utilizado em STIs tradicionais, comentado no capítulo 2, foi implementado nesta versão do MHITS na forma de base de dados textual, utilizando o modelo do estereótipo simplificado do aluno, veja capítulo 2, que considera apenas os níveis

iniciante, médio e avançado. Deve-se prever também, para uma próxima versão, a representação e implementação deste modelo no formato lógico, tarefa não trivial e que envolve níveis altos de complexidade por se tratar de um domínio multidisciplinar, devendo ser consultados profissionais das áreas de Pedagogia e Psicologia, além de Músicos e Professores de Música. Como comentado no capítulo 2, existem vários trabalhos nesta área, muitos ainda não concluídos, necessitando de um estudo focado e detalhado para o desenvolvimento deste modelo.

Nesta versão do MHITS, a base de conhecimento criada por [TEIX 97] foi modificada a fim de atender as especificações e detalhes relativos à comunicação com os módulos JAVA, necessitando assim da inclusão de comandos do próprio LPA-Prolog ++, principalmente para as tarefas de leitura e escrita de arquivos. Abaixo temos um diagrama que descreve o funcionamento da base Prolog na leitura do arquivo pergunta disponibilizado pelo Agente Comunicador (daemon) no diretório de perguntas, processamento da resposta e envio do arquivo resposta para o diretório correspondente, onde será lido pelo Agente Comunicador.



Complementando o diagrama acima, temos o esquema de funcionamento do Agente Comunicador, implementado na forma de daemon JAVA, que mantém-se em loop infinito durante a execução do MHITS, aguardando um comando de um dos módulos do MHITS, o Módulo Tutorial ou o Módulo de Perguntas, para então gerar o arquivo *querie* com a pergunta para a aplicação Prolog e colocá-lo no diretório de perguntas. Fica então aguardando a aplicação Prolog depositar o arquivo resposta no diretório de respostas, lê e apaga este arquivo, enviando a resposta para o applet que a solicitou.



As maiores dificuldades encontradas na implementação do MHITS foram achar um layout adequado para as telas da interface a fim de atender todas as necessidades de aprendizagem do usuário, trabalho a ser melhorado nas próximas versões, a escolha dos dados necessários para a modelagem do aluno e do professor, que perfil pedagógico seria utilizado pelo tutor e outros de menos importância. Mas a principal dificuldade, o que gerou um atraso muito grande no desenvolvimento e especificação do sistema, consequentemente gerando um

atraso na conclusão deste trabalho de dissertação, foi encontrar uma forma de fazer os agentes se comunicarem com a aplicação Prolog e manter esta última rodando a fim de executar esta comunicação, já que, por serem os ambientes Prolog interpretados, não temos a possibilidade da geração de arquivos executáveis. Mas a solução encontrada atende aos requisitos do sistema, necessitando apenas de um aperfeiçoamento no algoritmo, ou mesmo da linguagem utilizada no daemon JAVA, a fim de melhorar o desempenho do processo de comunicação e obtenção da resposta para o Aprendiz.

5.7. Conclusões

Neste capítulo apresentamos a descrição do MHITS – *Musical Harmony Intelligent Tutoring System*, um ambiente para o auxílio à aprendizagem da Harmonia Musical, abordando inicialmente uma descrição geral do sistema, o modelo de interação com telas e opções de navegação, seus requisitos básicos de dados e de plataforma de instalação, sua modelagem baseada em objetos e sua implementação baseada em agentes, onde os componentes do sistema foram discutidos e comentados individualmente.

O protótipo desenvolvido nesta versão foi apresentado, com ilustração de algumas telas e exemplos de interação a partir delas. Descrevemos ainda o ambiente de desenvolvimento, aspectos da linguagem escolhida para a implementação e porque ela foi escolhida, além de abordarmos detalhes de implementação e funcionalidade de cada módulo ou agente do nosso tutor.

Concluímos aqui que nossos objetivos foram alcançados para esta versão do sistema, necessitando ainda da aplicação de laboratórios de testes para captarmos as reações do usuário em relação à interação com o mesmo e assim obtermos sugestões e idéias para o seu melhoramento e aperfeiçoamento. Podemos já trazer a tona que o sistema necessita de uma melhora na parte gráfica e layouts das telas da interface, com a adição de mais recursos multimídia como vídeos e narrações, o que não foi explorado na versão apresentada aqui por termos priorizado as questões funcionais e de modelagem do sistema, inclusive porque a inclusão destes tipos de recursos multimídia geram um requerimento de infra-estrutura muito mais avançado, principalmente em relação à velocidade de transmissão via rede.

Gostaríamos ainda de salientar aqui que o código fonte do sistema e este trabalho como um todo poderá ser alcançado através do endereço http://www.krueger.com.br/CAMINHA/MHITS, contrariando os padrões normais de publicações nesta área, onde não temos muitos exemplos de código disponíveis para os novos pesquisadores, dificultando, às vezes, a resolução de problemas simples, que já foram solucionados anteriormente por outros pesquisadores e desenvolvedores.

Capítulo 6
Conclusões e Trabalhos Futuros

6.1. Considerações Finais

Neste trabalho foram abordados aspectos da IA, da Computação e Música, da Representação de Conhecimento em Harmonia Musical, do desenvolvimento de Sistemas Tutores Inteligentes, todos estes aplicados ao desenvolvimento de software educacional, mais especificamente para o ensino e aprendizagem de Harmonia Musical. Foi proposto um ambiente multimídia para auxiliar o ensino da Harmonia Musical, baseado em agentes.

A partir da análise dos trabalhos desenvolvidos em [COST 97] e [TEIX 97], além de apostilas e trabalhos desenvolvidos na área de ensino de Harmonia Musical, foi especificado e desenvolvido um ambiente tutor para utilização em salas de aula mas com recursos para a criação de programas de educação à distância, baseados na Internet.

A aplicação do ambiente pode ser feita em aulas práticas em laboratórios na própria escola de música, como também na casa do aluno, desde que ele tenha um computador multimídia a sua disposição. O objetivo é praticar e fixar melhor os conhecimentos adquiridos durante a aula, apresentada em sala ou via Internet. A utilização do ambiente permite que o aluno pratique os conceitos vistos e tenha a possibilidade de resolução de dúvidas simples acerca dos exercícios ou estruturas vistas em aula, através de um ambiente atrativo e adequado ao seu nível de aprendizagem.

A idéia do desenvolvimento de um sistema voltado para este domínio, além da ausência de sistemas tutores em Harmonia, veio da prática obrigatória exigida pelos professores de música no concernente ao estudo das teorias vistas em sala e a prática de exercícios, que nem sempre podem contar com um especialista humano para a resolução das dúvidas e para o acompanhamento do Aprendiz.

Como principal contribuição desta dissertação temos a especificação e desenvolvimento de um ambiente multimídia de ensino e aprendizagem em Harmonia Musical, com abordagem orientada a objetos, que utiliza uma representação do conhecimento voltada para a resolução de questões do usuário e exercícios do próprio sistema.

Como contribuições secundárias temos o desenvolvimento de uma interface multimídia, multiplataforma, com características de acesso via Internet, para este tipo de ambientes, incluindo o desenvolvimento e implementação de um mecanismo de comunicação entre applets JAVA e aplicações desenvolvidas em Prolog. Outra contribuição importante desenvolvida neste trabalho foi o estudo e implementação das características de divisão do domínio em subdomínios, exploração da comunicação entre agentes para a solução de questões do usuário, geradas através do tutorial ou através do módulo criado especificamente com este fim.

Observamos que a participação do especialista humano é fundamental para a aplicação correta do ambiente, sintonizando o seus objetivos pedagógicos com as capacidades e características da ferramenta, direcionando o aluno para o uso de novas tecnologias de aprendizagem, além de atuar como um gerenciador de todo o processo, observando o desempenho dos alunos através dos relatórios gerados pelo sistema.

A utilização de ferramentas multimídia por profissionais do ensino e pedagogia desperta o interesse por novas tecnologias e pelo uso de novas e mais atrativas fontes de conhecimento, que podem ser aplicadas paralelamente às atividades desenvolvidas em sala de aula.

Esta dissertação proporcionou, principalmente, a experiência da construção, especificação e implementação de um software educacional, com abordagem na metodologia orientada a objeto, incorporando características da IA, da Multimídia e de tecnologias Internet, em um domínio do conhecimento complexo mas mais do que interessante, o da Harmonia Musical.

6.2. Trabalhos Futuros

Como sugestão para trabalhos futuros temos a ampliação do sistema com o

desenvolvimento de alguns módulos que podem trazer mais interatividade, usabilidade e principalmente uma personalização inteligente do sistema a partir de cada usuário. Como sugestão, propomos o desenvolvimento de:

- uma modelagem do aluno que permita a adaptação contínua do sistema ao aprendiz, de forma a contemplar os aspectos de interface adaptativa contemplados no MATHEMA [COST 97],

- uma ampliação da base de conhecimento em Harmonia Musical, e

- mecanismos para utilização de instrumentos musicais que obedecem o padrão MIDI (*Musical Instruments Digital Interface*), como teclados, guitarras, violinos e outros, diretamente ligados à interface do sistema. A base de conhecimento já prevê esta característica, o que torna mais fácil este desenvolvimento.

Referências Bibliográficas

[ALEX 96] ALEXE, C.; GECSEI, J.; *A Learning Environment for the Surgical Intensive Care Unit*; Procc. of ITS96 - Frasson, Gauthier, Lesgold (eds), Montreal, pp439-446, jun 1996.

[ANDE 90] ANDERSON, J.; BOYLE, C.; CORBETT, A.; LEWIS, M.; *Cognitive Modeling and Intelligent Tutoring*; Artificial Intelligence, Vol 42, nº 1, pp 7-49, 1990.

[ASAN 91] ASANOME, C. ; *Sistemas Tutoriais Inteligentes: um Estudo*; Monografia de final de curso de Engenharia de Software, COPPE/UFRJ, 1991.

[BERT 94] BERTLS, K.; *A Dynamic View on Cognitive Student Modelling in Computer Programming*; Journal of Artificial Intelligence in Education, Vol 5, nº 1, pp 85-105, 1994.

[BRIS 79] BRISOLLA, C. M.; *Princípios de Harmonia Funcional*, Editora Novas Metas, São Paulo, SP, 1979.

[BRUS 93] BRUSILOVSKY, P.*; Student as User: Toward an Adaptative Interface for an Intelligent Learning Environment*; Proccedings of AI-ED93 World Conference on Artificial Intelligence in Education, pp 386-393, 1993.

[CAMI 96] CAMINHA, A. de O., Teixeira, L. de M., Aleixo, F. A., Costa, E. de B., Ferneda, E.; *A Noção de Ponto de Vista Aplicada à Representação de Eventos Musicais*, Anais da VI Semana de Informática da UFBA - SEMINFO'96, pp. 67-78, Salvador, BA, 1996.

[CARV 94] CARVALHO, R.; *Teoria Musical III: Parâmetro Altura*, Fundação Cultural Monsenhor Chaves, Teresina, PI, 1994.

[CAVA 91] CAVALLO, D.; *New Meaning for Learning*; Anais do VII Simpósio Brasileiro de Inteligência Artificial, SBC, 1991.

[CHED 85] CHEDIAK, A.; *Harmonia & Improvisação*, Lumiar Editora, Rio de Janeiro, RJ, 1985

[CORR 93] CORREDOR, M. V.; *La Inteligencia Artificial y la Educacion: lo aprendido y las futuras acciones*; Informática Educativa, Colômbia, vol 6, n. 3, pp 235-242, 1993.

[COST 97a] COSTA, E. B., TEIXEIRA, L. M., FERNEDA, E.*; Um Sistema Tutor Inteligente Multi-Agente em Harmonia Musical*; Anais do VIII Simpósio Brasileiro de Informática na Educação, São José dos Campos - SP, novembro 1997.

[COST 97b] COSTA, E. B.; *Um Modelo de Ambiente Interativo de Aprendizagem Baseado numa Arquitetura Multi-Agentes*; Tese de Doutorado em Engenharia Elétrica, Universidade Federal da Paraíba - Campus II, dezembro 1997.

[COST 97c] COSTA, E. B., LOPES, M. A., FERNEDA, E.*; Mathema: A learning environment based on a multi-agent architecture;* In Proceedings of SBIA'95, Springer-Verlager, 1995.

[COST 98a] COSTA, R. M. E. M. da; ROCHA, A.R.C. da; SANTOS, N. dos; WERNECK, V.M.B.; *Desenvolvimento de Sistemas Tutores Inteligentes: Questões e Perpectivas*, COPPE Sistemas e Computação, UFRJ, 1998.

[COST 98b] COSTA, R. M. E. M. da; WERNECK, V.M.B.; *Sistemas Tutoriais: Aplicações das Tecnologias de Hipermídia e de Inteligência Artificial em Educação*, COPPE Sistemas e Computação, UFRJ, 1998.

[DANN 94] Dannenberg, R. B.; *Music Representation Issues, Techniques, and Systems*, Computer Music Journal, Vol. 17, nº 3, pp. 20-30, MIT Press, EUA, 1994.

[ECKE 97] ECKEL, B.; *Thinking in JAVA*, http://www.BruceEckel.com/, CodeGuru, 1997.

[ELIO 96] ELIOT, C.; WOOLF, B.; *Iterative Development and Validation of a Simulation-Based Medical Tutor*; Procc. of ITS96 - Frasson, Gauthier, Lesgold (eds), Montreal, pp 540-549, jun 1996.

[ELOR 96] ELORRIAGA, J.; FERNÁNDEZ-CASTRO, I.; *The HSIIP Approach. An Extension for a Teacher's Apprentice*; Procc. of ITS96 - Frasson, Gauthier, Lesgold (eds),Montreal, pp 401-410, jun 1996.

[FARJ 93] FARJADO, F.R.; *Que puede aportar la Inteligência Artificial al desarrollo de la Informática Educativa ?*; Tese de Mestrado - COPPE Sistemas e Computação, UFRJ, 1995.

[FISC 90] FISCHETTI, E.; GISOLF, A.; *From Computer-Aided Instruction to Intelligent Tutoring Systems*; Educational Tecnology, pp 7-17, agosto de 1990.

[GEIS 92] GEISSMAN, J.; SCHULTZ, R.; *Verification & Validation of Expert Systems, Knowledg-based Systems: Fundamentals and Tool*s, eds Garcia e Chien, IEEE Computer Society Press, 1992.

[GLAN 95] GLANZMANN, J.H.; *Expert Piano: Um Ambiente de Auxílio a Aprendizagem Musical*; Tese de Mestrado - COPPE Sistemas e Computação, UFRJ, 1995.

[HIND 86] HINDEMITH, P.; *Harmonia Tradicional*, Irmãos Vitale, São Paulo, 1986.

[HONI 93] HONING, H.; *Issues in the Representation of Time and Structure in Music*, Contemporary Music Review, nº 9, 1993. (ftp://mars.let.uva.nl/ honing/PAPERS/H-93-B. HTML).

[JEPS 97] JEPSON, B.; *Programando BANCO de DADOS em JAVA*, Makron Books, 1997.

[LEMA 96] LEMAY, L., PERKINS, C. L.; *Teach Yourself JAVA in 21 Days*, Sams.net Publishing, 1996.

[LIND 98] LINDEN, P. van der; *Just JAVA*, Guia Autorizado SUNSOFT Press, Sun Mycrosystems, Makron Books, 1998.

[MAGA 95] MAGALHÃES NETTO, J.; *Um Tutor Inteligente para o Ensino de Xadrez*; Tese de Mestrado do Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ, 1995.

[MARK 93] MARK, M.; GREER, J.; *Evaluation Methodologies for Intelligent Tutoring Systems*; Journal of Artificial Intelligence in Education, 4(2/3), pp 129-153, 1993.

[MCCA 94] McCALLA, G.*; Artificial Intelligence and Educational Technology: a Natural Synergy*; Educational Multimedia an Hypermedia Proccedings of ED-MEDIA 94, Vancouver, pp 47-49, junho 1994.

[MINS 85] MINSKY, M.; *The Society of Mind*, Simon and Schuster, New York (EUA), 1985.

[MOSS 95] MOSS, C.; *Prolog++: The Power of Object-Oriented and Logic-Programming*; Addison-Wesley Publishing Company, Cambridge, Inglaterra, 1995.

[NUNE 93] NUNES, M. das G.; TURINE, M.; MALTEPI, M.; HASEGAWA, R.; *Uso de Hipertexto/Hipermídia em Sistemas Tutores Inteligentes*; Notas Didáticas do Instituto de Ciências Matemáticas de São Carlos, SP, nº 9, março de 1993.

[PERE 85] PÉREZ, M.; *Diccinario de la Música y los Músicos*, Ediciones ISTMO, Madrid, Espanha, 1985.

[PINT 95] PINTO, S.C.; *M-Assiste: Um Meta-Assistente Adaptativo para Suporte à Navegação em Documentos Hipermídia*; Tese de Mestrado do Programa de Engenharia de Sistemas e Computação, COPPE-UFRJ, dezembro de 1995.

[POPE 91] POPE, S. T.; *The Well-Tempered Object: Musical Applications of Objetc-Oriented Software Technology*, MIT Press, EUA, 1991.

[PRIO 85] PRIOLLI, M. L. de M.; *Princípios Básicos da Música para a Juventude*, Editora Casa Oliveira de Músicas LTDA., Rio de Janeiro, RJ, 1985.

[RATT 92] RATTON, M. B.; *MIDI - Guia Básico de Referência*, Editora Campus, Rio de Janeiro, RJ, 1992.

[REGI 94] REGIAN, J.W.; SHUTE, V.; *Evaluating Intelligent Tutoring Systems*; Technology Assessment in Education and Training, Baker e O'Neil (Eds), Lawrence Erlbaum Associates, New Jersey, pp. 79-96, 1994.

[REIN 93] REINHARDT, B.*;* SCHEWE, S.A.*; Intelligent Tutoring Systems*; Proccedings of AI-ED95 World Conference on Artificial Intelligence in Education, Washington, 1995.

[RICH 94] RICH, E., KNIGHT, K.; *Inteligência Artificial*; Makron Books, São Paulo, 1994.

[ROBE 97] ROBERTS, S., HELLER, P.; *JAVA 1.1 Certification Study Guide*, SYBEX Inc., 1997

[SCHO 79] SCHONBERG, A.; *Tratado de Armonía*, Real Musical, Madri, Espanha, 1979.

[SHUT 94] SHUT, V.; *Regarding the I in ITS: Student Modeling*; Educational Multimedia and Hypermedia Proccedings of ED-MEDIA 94, Canadá, junho de 1994.

[SIEB 97] SIEBRA, S. A., Tedesco, P. A., Ramalho, G. L., Barros, F. A., Souza, F. F.*;*

Ambiente Inteligente de ensino Baseado em uma arquitetura Multi-Agentes; Anais do VIII Simpósio Brasileiro de Informática, 1997

[TEDE 97] TEDESCO, P. A., BARROS, F. A., SOUZA, F. F.; *SEI – Sistema de Ensino Inteligente*, Anais do VIII Simpósio Brasileiro de Informática na Educação, São José dos Campos - SP, novembro 1997.

[TEIX 97] TEIXEIRA, L. M.*; Da Representação do Conhecimento Musical ao Esboço de uma Sociedade de Agentes em Harmonia*; Dissertação de Mestrado em Informática. Universidade Federal da Paraíba - Campus II, setembro 1997.

[VASE 95] VASEY, P.; *LPA-Prolog++ 2.0, Programmer's Reference*; Logic Programming Associates Ltd., Londres, Inglaterra, 1995.

[VICC 90] VICCARI, R., MOUSSALE, N.*; Tutores Inteligentes para o Ensino de Linguagem PROLOG*; Anais do I Simpósio Brasileiro de Informática na Educação, 1990.

[VICC 92] VICCARI, R.; OLIVEIRA, F.; *Sistemas Tutores Inteligentes*; Monografia do Instituto de Informática – UFRGS, setembro de 1992

[VICC 93] VICCARI, R.*; Inteligência Artificial e Educação: Indagações Básicas*; Anais do IV Simpósio Brasileiro de Informática na Educação, pp 207-216, 1993.

[VICC 96] VICCARI, R., Giraffa, M. M. L.*; Sistemas Tutores Inteligentes: Abordagem Tradicional x Abordagem de Agentes*; XIII Simpósio Brasileiro de Inteligência Artificial, Sociedade Brasileira de Computação, Tutorial T6, 1996.

[WARR 93] WARREN, K.C.; GOODMAN, B.A.; MACIOROWSKI, S.M.; *A Software Architecture for Intelligent Tutoring Systems*; Proccedings of AI-ED93 World Conference on Artificial Intelligence in Educational, Edinburgh, pp 50-57, 1993.

[WENG 87] WENGER, E.; *Artificial Intelligence and Tutoring Systems – Computational and Cognitive Approaches to the Communication of Knowledge*; Morgan Kaufmann, Los Altos, EUA, 1987.

[WERN 95] WERNECK, V. B.; *Ambientes de Desenvolvimento de Sistemas Baseados em Conhecimento*, Tese de Doutorado do Programa de Engenharia de Sistemas e Computação, COPPE-UFRJ, mar 1995.

[WINK 93] WINKELS, R.; BREUKER, J.; *Rational Reconstruction of Diagnostic Expertise*; *KADS: A Principled Approach to KBS Development*, Scheriber, Wielling and Breuker (ed), Academic Press, pp 314-336, 1993.

[WOOL 88] WOOLF, B.; *Intelligence Tutoring Systems*; In: Exploring Artificial Intelligence: Survey Talks from Natural Conferences on Artificial Intelligence, S. Howard (Ed). Morgan Kaufmann, 1988.

Apêndice

Código JAVA (Agente Interface)

```
/* Arquivo: AgenteInterface.java
 * Modulo do: MHITS - Musical Harmony Intelligent Tutoring System
 * Funcao: Gerar telas de acordo com o nivel do usuario, Controlar Login,
           Ativar o Agente Tutor, Ativar o cadastro
 * Autor: Adriano de O. Caminha            (caminha@dsc.ufpb.br)
 * Colaboracao: Denis Augusto A. de Souza   (denis_souza@yahoo.com)

 * Data: 04/02/2000       Hora: 16:16
 * Tipo: Applet
 * Versao do JDK usadao: 1.1.8
 * Melhorias:

 * Atualizacoes:

        05/02/2000      Terminado o metodo init, paint.
        06/02/2000      Terminada a primeira, segunda e terceira tela (login).
        11/02/2000      Incluido o parametro daemon_url no applet para
comunicacao.
                        Incluido a passagem de parametro sobre o nivel do usuario
                        para o cadastro.

 * Metodos Usados:
        main    Metodo Principal
        action  Elaborar uma acao especifica ao ser gerado um evento
        paint   Pinta objetos graficos na tela

        Construtor: AgenteInterface

 * Objetos Usados:

 * imports Usados:

        java.awt.*        Pacote de classes usados para a criacao de objetos
                          graficos independente de plataforma (inerente ao Java)
        java.awt.event.*  Eventos para tratamento do mouse e objetos da interface
java
        java.applet.*     Classes para o tratamento de aplicacoes applets
        Msql*             Pacote de classes para o trabalho com o Msql
        TextField2        Classe para limitar e controlar o tipo de dados
inseridos
                          via teclado (inteiros, strings...)
        Bib_MHITS         Biblioteca do sistema MHITS
*/

 // Padrao
 import java.awt.*;
 import java.awt.event.*;
 import java.applet.*;

 //import sec2;
```

```
 import Bib_MHITS;
 import AgenteAluno_Cadastro;
 import TextField2;
 import ImageCanvas;

 //Para acesso ao Msql Server
 import MsqlException;
 import MsqlRow;
 import MsqlFieldDesc;
 import MsqlData;
 import MsqlResult;
 import Msql;

public class AgenteInterface extends Applet implements
        KeyListener,ActionListener{

        // ****************************************************************
        // Objetos Locais
        String SQL_HostName, Msql_Usuario;        // SQL tratamento
        String Guarda_Nome_Usuario, Guarda_Nivel_Usuario,
               Guarda_Login_Codigo_Usuario;
        String Daemon_URL;

        // Guarda a tecla que foi digitada
        char Guarda_Caracter_Digitado = ' ';
        String Caracter_Digitado_Flag = new String();

        // Define um objeto para acessar a biblioteca
        Bib_MHITS Biblioteca_MHITS = new Bib_MHITS();

        // Para a geracao de um beep sonoro
        // Recupera o "kit de ferramentas"padrao para uso
        Toolkit Beep_Toolkit = Toolkit.getDefaultToolkit();

        // *********** Msql Objetos ***********
        Msql msql = new Msql();          // Objeto para tratar da base de dados
        MsqlResult result;               // Objeto para capturar o resultado de
um comando

        // ****************************************************************

        // *********** Objetos da Interface de Tela ********

        // Posiconamento de Objetos
        int Posicao_X_Button = 80;
        int Posicao_Y_Button = 163;
        int Espaco_Y_Botoes = 58;

        int Posicao_X_Login = 252;
        int Posicao_Y_Login = 120;

        // Labels
        Label Linha_Msg;

        // Botoes
        Button Botao_Sobre_o_MHITS, Botao_Como_Usar_o_MHITS, Botao_Entrar;
        Button Botao_Cancela, Botao_OK, Botao_Login_Anonimo;
        Button Botao_Tutorial, Botao_Cadastro;

        // Define o Tamanho do Titulo
        Font Fonte_TextField_Login = new Font("Serif", Font.PLAIN, 20);
        Font Fonte_Labels = new Font("Serif", Font.PLAIN, 12); // Font para o p
        Font Fonte_Linha_Msg = new Font("Serif", Font.BOLD, 14); // Font para o p
```

```
        Font Fonte_Menu = new Font("Serif", Font.PLAIN, 12); // Font para o paint
        Font Fonte_Botoes = new Font("SansSerif", Font.BOLD, 16); // Font para o
paint
        Font Fonte_Botoes_Tela3 = new Font("SansSerif", Font.BOLD, 24); // Font
para o paint

        Color Cor_BackGround = SystemColor.control;
        Color Cor_Texto = SystemColor.controlText;
        Color Cor_BackGround_TextField = Color.yellow;
        Color Cor_Texto_TextField = new Color(51,51,255);
        Color Cor_Texto_Linha_Msg = Color.white;
        Color Cor_BackGround_Linha_Msg = new Color(132,44,96);
        Color Cor_BackGround_Botao = new Color(132,44,96);
        Color Cor_Texto_Botao = Color.white;

        // ****************************************************************

        // *********** Objetos para acessar outros Modulos ********
        AgenteAluno_Cadastro Agente_Aluno_Cadastro;

        // ****************************************************************

        // ********************** TextFields ******************************
        TextField2 Login_Aluno_TextField, Senha_Aluno_TextField;

        // ********************** Imagens *********************************
        ImageCanvas Imagem_Tela_Entrada, Imagem_Login, Imagem_Menu_Entrada;

        // ****************************************************************

        // Init
        public void init(){

                // Referentes ao appet
                super.init();

                setLayout(null);
                addNotify();
                setBackground(Color.white);

                // Nome do SQL Host
                // Capturando os parametros introduzidos no start do applet
                SQL_HostName = getParameter("sql_host");
                Daemon_URL = getParameter("daemon_url");

                // Define qual o usuario definido no msql.acl para ter acesso
                Msql_Usuario = Biblioteca_MHITS.PegaUser();

                // Instancia os objetos para coleta de dados dos usuarios
                Guarda_Nome_Usuario = new String();
                Guarda_Nivel_Usuario = new String();
                Guarda_Login_Codigo_Usuario = new String();

                // ********* Carrega Imagens ***********
                // Carrega o tela de entrada
                Imagem_Tela_Entrada = new
ImageCanvas(".\\imagens\\bg\\entrada_tela.gif");
                Imagem_Tela_Entrada.setSize(532,399);
                Imagem_Tela_Entrada.setLocation(0,0);
                Imagem_Tela_Entrada.setVisible(true);

                // Carrega o tela de login
                Imagem_Login = new ImageCanvas(".\\imagens\\bg\\login_tela.gif");
```

```
                Imagem_Login.setSize(532,399);
                Imagem_Login.setLocation(0,0);
                Imagem_Login.setVisible(false);

                // Carrega o tela do menu de entrada
                Imagem_Menu_Entrada = new
ImageCanvas(".\\imagens\\bg\\menu_entrada_tela.gif");
                Imagem_Menu_Entrada.setSize(532,399);
                Imagem_Menu_Entrada.setLocation(0,0);
                Imagem_Menu_Entrada.setVisible(false);

                // ********** Objetos da Interface de Tela ********
                // ****  Labels  *****
                Linha_Msg = new Label("Complete os campos
necessários",Label.CENTER);
                Linha_Msg.reshape(0,372,505,25);
                Linha_Msg.setForeground(Cor_Texto_Linha_Msg);
                Linha_Msg.setBackground(Cor_BackGround_Linha_Msg);
                Linha_Msg.setFont(Fonte_Linha_Msg);

                // Primeira tela (Entrada)
                // Botoes
                Botao_Sobre_o_MHITS = new Button("Sobre o MHITS");

Botao_Sobre_o_MHITS.setBounds(Posicao_X_Button,Posicao_Y_Button,177,36);
                Botao_Sobre_o_MHITS.setForeground(Cor_Texto_Botao);
                Botao_Sobre_o_MHITS.setBackground(Cor_BackGround_Botao);
                Botao_Sobre_o_MHITS.setCursor(new Cursor(Cursor.HAND_CURSOR));
                Botao_Sobre_o_MHITS.setFont(Fonte_Botoes);
                Botao_Sobre_o_MHITS.addActionListener(this);
                Botao_Sobre_o_MHITS.setVisible(true);

                Botao_Como_Usar_o_MHITS = new Button("Como Usar o MHITS");

Botao_Como_Usar_o_MHITS.setBounds(Posicao_X_Button,Posicao_Y_Button+Espaco_Y_Boto
es-1,177,36);
                Botao_Como_Usar_o_MHITS.setForeground(Cor_Texto_Botao);
                Botao_Como_Usar_o_MHITS.setBackground(Cor_BackGround_Botao);
                Botao_Como_Usar_o_MHITS.setCursor(new
Cursor(Cursor.HAND_CURSOR));
                Botao_Como_Usar_o_MHITS.setFont(Fonte_Botoes);
                Botao_Como_Usar_o_MHITS.addActionListener(this);
                Botao_Como_Usar_o_MHITS.setVisible(true);

                Botao_Entrar = new Button("Entrar");

Botao_Entrar.setBounds(Posicao_X_Button,Posicao_Y_Button+(2*Espaco_Y_Botoes),177,
36);
                Botao_Entrar.setForeground(Cor_Texto_Botao);
                Botao_Entrar.setBackground(Cor_BackGround_Botao);
                Botao_Entrar.setCursor(new Cursor(Cursor.HAND_CURSOR));
                Botao_Entrar.setFont(Fonte_Botoes);
                Botao_Entrar.addActionListener(this);
                Botao_Entrar.setVisible(true);

                // Segunda tela (Login)
                Botao_Cancela = new Button("Cancela");

Botao_Cancela.setBounds(Posicao_X_Button+168,Posicao_Y_Button+(Espaco_Y_Botoes-
20),100,36);
                Botao_Cancela.setForeground(Cor_Texto_Botao);
                Botao_Cancela.setBackground(Cor_BackGround_Botao);
                Botao_Cancela.setCursor(new Cursor(Cursor.HAND_CURSOR));
```

```
                Botao_Cancela.setFont(Fonte_Botoes);
                Botao_Cancela.addActionListener(this);
                Botao_Cancela.setVisible(false);

                Botao_OK = new Button("OK");

Botao_OK.setBounds(Posicao_X_Button+288,Posicao_Y_Button+(Espaco_Y_Botoes-
20),100,36);
                Botao_OK.setForeground(Cor_Texto_Botao);
                Botao_OK.setBackground(Cor_BackGround_Botao);
                Botao_OK.setCursor(new Cursor(Cursor.HAND_CURSOR));
                Botao_OK.setFont(Fonte_Botoes);
                Botao_OK.addActionListener(this);
                Botao_OK.setVisible(false);

                Botao_Login_Anonimo = new Button("Login Anônimo");

Botao_Login_Anonimo.setBounds(Posicao_X_Button+210,Posicao_Y_Button+(3*Espaco_Y_B
otoes-23),177,37);
                Botao_Login_Anonimo.setForeground(Cor_Texto_Botao);
                Botao_Login_Anonimo.setBackground(Cor_BackGround_Botao);
                Botao_Login_Anonimo.setCursor(new Cursor(Cursor.HAND_CURSOR));
                Botao_Login_Anonimo.setFont(Fonte_Botoes);
                Botao_Login_Anonimo.addActionListener(this);
                Botao_Login_Anonimo.setVisible(false);

                // Terceira tela (menu_entrada -> tutorial, cadastro)
                Botao_Tutorial = new Button("Tutorial");
                Botao_Tutorial.setBounds(Posicao_X_Button+7,Posicao_Y_Button-
(24),179,40);
                Botao_Tutorial.setForeground(Cor_Texto_Botao);
                Botao_Tutorial.setBackground(Cor_BackGround_Botao);
                Botao_Tutorial.setCursor(new Cursor(Cursor.HAND_CURSOR));
                Botao_Tutorial.setFont(Fonte_Botoes_Tela3);
                Botao_Tutorial.addActionListener(this);
                Botao_Tutorial.setVisible(false);

                Botao_Cadastro = new Button("Cadastro");

Botao_Cadastro.setBounds(Posicao_X_Button+7,Posicao_Y_Button+(46),179,40);
                Botao_Cadastro.setForeground(Cor_Texto_Botao);
                Botao_Cadastro.setBackground(Cor_BackGround_Botao);
                Botao_Cadastro.setCursor(new Cursor(Cursor.HAND_CURSOR));
                Botao_Cadastro.setFont(Fonte_Botoes_Tela3);
                Botao_Cadastro.addActionListener(this);
                Botao_Cadastro.setVisible(false);


                // **** TextField ****
                Login_Aluno_TextField = new TextField2("",30,30,"String");

Login_Aluno_TextField.reshape(Posicao_X_Login,Posicao_Y_Login+1,214,26);
                Login_Aluno_TextField.setForeground(Cor_Texto_TextField);
                Login_Aluno_TextField.setBackground(Cor_BackGround_TextField);
                Login_Aluno_TextField.setFont(Fonte_TextField_Login);
                Login_Aluno_TextField.addKeyListener(this);
                Login_Aluno_TextField.setVisible(false);

                Senha_Aluno_TextField = new TextField2("",8,8,"String");

Senha_Aluno_TextField.reshape(Posicao_X_Login,Posicao_Y_Login+40,214,26);
                Senha_Aluno_TextField.setForeground(Cor_Texto_TextField);
                Senha_Aluno_TextField.setBackground(Cor_BackGround_TextField);
```

```
                Senha_Aluno_TextField.setFont(Fonte_TextField_Login);
                Senha_Aluno_TextField.addKeyListener(this);
                Senha_Aluno_TextField.setEchoChar('*');
                Senha_Aluno_TextField.setVisible(false);

                // **** Adiciona os Objetos *****
                // Labels
                add(Linha_Msg);

                // TextField
                add(Login_Aluno_TextField);
                add(Senha_Aluno_TextField);

                // Botoes
                add(Botao_Sobre_o_MHITS);
                add(Botao_Como_Usar_o_MHITS);
                add(Botao_Entrar);

                add(Botao_Cancela);
                add(Botao_Login_Anonimo);
                add(Botao_OK);

                add(Botao_Tutorial);
                add(Botao_Cadastro);

                // imagens
                add(Imagem_Tela_Entrada);
                add(Imagem_Login);
                add(Imagem_Menu_Entrada);

                // Ativa a primeira tela
                Imagem_Tela_Entrada.setVisible(true);

        } // end init

        // ******** Metodos para trabalhar como teclado **********
        // ******** Fazendo a acentuacao *********
        public void keyTyped(KeyEvent e){

                // Objetos Locais
                char Caracter_Digitado_Char = e.getKeyChar();

                if (Caracter_Digitado_Flag.equals("Invalido")){

                        Linha_Msg.setText("Este caracter é reservado para uso
interno ao sistema.");
                        Beep_Toolkit.beep();
                        e.setKeyChar('?');
                }

                // acento agudo
                if (Guarda_Caracter_Digitado == ''')
                        switch (Caracter_Digitado_Char){

                                case 'a': // A acentuado

                                        e.setKeyChar('á');
                                        Guarda_Caracter_Digitado = ' ';
                                        break;

                                case 'A': //
                                        e.setKeyChar('Á');
                                        Guarda_Caracter_Digitado = ' ';
```

```
                        break;

                case 'e': //
                        e.setKeyChar('é');
                        Guarda_Caracter_Digitado = ' ';
                        break;

                case 'E': //
                        e.setKeyChar('É');
                        Guarda_Caracter_Digitado = ' ';
                        break;

                case 'i': //
                        e.setKeyChar('í');
                        Guarda_Caracter_Digitado = ' ';
                        break;

                case 'I': //
                        e.setKeyChar('Í');
                        Guarda_Caracter_Digitado = ' ';
                        break;

                case 'o': //
                        e.setKeyChar('ó');
                        Guarda_Caracter_Digitado = ' ';
                        break;

                case 'O': //
                        e.setKeyChar('Ó');
                        Guarda_Caracter_Digitado = ' ';
                        break;

                case 'u': //
                        e.setKeyChar('ú');
                        Guarda_Caracter_Digitado = ' ';
                        break;

                case 'U': //
                        e.setKeyChar('Ú');
                        Guarda_Caracter_Digitado = ' ';
                        break;

                case 'c': //
                        e.setKeyChar('ç');
                        Guarda_Caracter_Digitado = ' ';
                        break;

                case 'C': //
                        e.setKeyChar('Ç');
                        Guarda_Caracter_Digitado = ' ';
                        break;
        }

// acento grave
if (Guarda_Caracter_Digitado == '`')
        switch (Caracter_Digitado_Char){

                case 'a': // A acentuado

                        e.setKeyChar('à');
                        Guarda_Caracter_Digitado = ' ';
                        break;
```

```
                        case 'A': //
                                e.setKeyChar('À');
                                Guarda_Caracter_Digitado = ' ';
                                break;

                }


        // til
        if (Guarda_Caracter_Digitado == '~')
                switch (Caracter_Digitado_Char){

                        case 'a': // A acentuado

                                e.setKeyChar('ã');
                                Guarda_Caracter_Digitado = ' ';
                                break;

                        case 'A': //
                                e.setKeyChar('Ã');
                                Guarda_Caracter_Digitado = ' ';
                                break;

                        case 'o': // A acentuado

                                e.setKeyChar('õ');
                                Guarda_Caracter_Digitado = ' ';
                                break;

                        case 'O': //
                                e.setKeyChar('Õ');
                                Guarda_Caracter_Digitado = ' ';
                                break;
                }

        // circunflexo
        if (Guarda_Caracter_Digitado == '^')
                switch (Caracter_Digitado_Char){

                        case 'e': // A acentuado

                                e.setKeyChar('ê');
                                Guarda_Caracter_Digitado = ' ';
                                break;

                        case 'E': //
                                e.setKeyChar('Ê');
                                Guarda_Caracter_Digitado = ' ';
                                break;

                }

}

public void keyPressed(KeyEvent e){

        // Objetos Locais
        char Caracter_Digitado_Char = e.getKeyChar();

        // Simbolos nao permitidos
        switch (Caracter_Digitado_Char){

                case '@': //
```

```
        Caracter_Digitado_Flag = "Invalido";
        break;

case '{': //

        Caracter_Digitado_Flag = "Invalido";
        break;

case '}': //

        Caracter_Digitado_Flag = "Invalido";
        break;

case '[': //

        Caracter_Digitado_Flag = "Invalido";
        break;

case ']': //

        Caracter_Digitado_Flag = "Invalido";
        break;

case '!': //

        Caracter_Digitado_Flag = "Invalido";
        break;

case '<': //

        Caracter_Digitado_Flag = "Invalido";
        break;

case '%': //

        Caracter_Digitado_Flag = "Invalido";
        break;

case '*': //

        Caracter_Digitado_Flag = "Invalido";
        break;

case '&': //

        Caracter_Digitado_Flag = "Invalido";
        break;

case '+': //

        Caracter_Digitado_Flag = "Invalido";
        break;

case '=': //

        Caracter_Digitado_Flag = "Invalido";
        break;

case '_': //

        Caracter_Digitado_Flag = "Invalido";
        break;
```

```
                        case '>': //

                                Caracter_Digitado_Flag = "Invalido";
                                break;

                        default: //
                                Caracter_Digitado_Flag = "Valido";
                }

                if (Caracter_Digitado_Char == ''' ||  Caracter_Digitado_Char ==
'`'
                   || Caracter_Digitado_Char == '~' || Caracter_Digitado_Char ==
'^')
                        Guarda_Caracter_Digitado = Caracter_Digitado_Char;

        }

        public void keyReleased(KeyEvent e){

                //System.out.println("release:"+e);

        }

        /* ************************************************************
        *
        * Funcao: Elaborar o tratamento de eventos
        * Ao ser gerado um evento o applet olha este metodo
        * Metodo Inerente a interface de um Applet
        */

        public boolean action (Event evento, Object objeto){

                // Sai do metodo
                return(true);
        }

        // Coleta a acao dos botoes
        public void actionPerformed(ActionEvent evt){

                if (evt.getSource() ==  Botao_Entrar){

                        // Desabilita os botoes anteriores
                        Botao_Sobre_o_MHITS.setVisible(false);
                        Botao_Como_Usar_o_MHITS.setVisible(false);
                        Botao_Entrar.setVisible(false);

                        Linha_Msg.setText("Contactando o sistema...");

                        // Adiciona os novos botoes
                        Botao_Cancela.setVisible(true);
                        Botao_OK.setVisible(true);
                        Botao_Login_Anonimo.setVisible(true);

                        // Adiciona os Textfields
                        Login_Aluno_TextField.setVisible(true);
                        Senha_Aluno_TextField.setVisible(true);

                        Imagem_Login.setVisible(true);
                        Imagem_Tela_Entrada.setVisible(false);

                        Linha_Msg.setText("Digite os dados para o entrada no
sistema.");
```

```
                }

                if (evt.getSource() ==  Botao_Cancela){

                        // Desabilita os botoes anteriores
                        Botao_Cancela.setVisible(false);
                        Botao_OK.setVisible(false);
                        Botao_Login_Anonimo.setVisible(false);

                        Login_Aluno_TextField.setVisible(false);
                        Senha_Aluno_TextField.setVisible(false);

                        Imagem_Login.setVisible(false);

                        Linha_Msg.setText("Contactando o sistema...");

                        // Adiciona os novos botoes
                        Botao_Sobre_o_MHITS.setVisible(true);
                        Botao_Como_Usar_o_MHITS.setVisible(true);
                        Botao_Entrar.setVisible(true);

                        Imagem_Tela_Entrada.setVisible(true);

                        Linha_Msg.setText("Elabore a sua escolha.");
                }

                if (evt.getSource() ==  Botao_OK){

                        // Define novos objetos para o trabalho
                        String Login_Digitado_String =
Login_Aluno_TextField.getText();
                        String Senha_Digitada_String =
Senha_Aluno_TextField.getText();

                        Linha_Msg.setText("Aguarde a verificação dos dados...");

                        if ( (Login_Digitado_String.equals("")) ||
                             (Senha_Digitada_String.equals("")) ){

                                Linha_Msg.setText("Algum campo está vazio. Tente
novamente...");
                                Beep_Toolkit.beep();

                        } else {

                                if (!Porteiro("Usuario_Registrado", "aluno")){

                                        Linha_Msg.setText("Primeira etapa da
verificação terminada. Aguarde...");

                                        if (!Porteiro("Usuario_Registrado",
"professor")){

                                                Linha_Msg.setText("Dados
inválidos. Tente novamente.");
                                                Beep_Toolkit.beep();

                                        } else {

                                                Linha_Msg.setText("Entrada no
sistema autorizada.");
                                                Monta_Menu_Entrada_Tela();
                                        }
```

```
                                } else
                                        Monta_Menu_Entrada_Tela();
                        }
                }

                if (evt.getSource() ==  Botao_Login_Anonimo){

                        Linha_Msg.setText("Aguarde o processamento...");

                        if (Porteiro("Usuario_Anonimo","")){

                                Linha_Msg.setText("Entrada no sistema
autorizada.");
                                Monta_Menu_Entrada_Tela();
                        }
                }

                if (evt.getSource() ==  Botao_Cadastro){

                        Linha_Msg.setText("Aguarde o processamento...");

                        Agente_Aluno_Cadastro = new
AgenteAluno_Cadastro(SQL_HostName, Msql_Usuario,
Guarda_Nivel_Usuario,Guarda_Login_Codigo_Usuario);
                        Agente_Aluno_Cadastro.setVisible(true);

                        Linha_Msg.setText("Cadastro de usuários pronto para
operação.");
                }

        }

        // Porteiro para a elaboracao da tarefa de Login
        public boolean Porteiro(String Tipo_de_Login_String, String
Tabela_String){

                // Objetos locais
                String Login_Digitado_String = Login_Aluno_TextField.getText();
                String Senha_Digitada_String = Senha_Aluno_TextField.getText();
                String Msql_Comando = new String();
                String row[];

                boolean Encontrou_Registro_Boolean = false;

                // Usuario registrado
                if (Tipo_de_Login_String.equals("Usuario_Registrado")){

                            try{

                                msql.Connect(SQL_HostName,Msql_Usuario);
                                msql.SelectDB("mhits");

                                if (Tabela_String.equals("aluno"))
                                        Msql_Comando = "select login,nome from
"+Tabela_String+
                                                " where login clike
'"+Login_Digitado_String+
                                                "' and password
='"+Senha_Digitada_String+"'";
                                else
                                        Msql_Comando = "select login,nome,codigo
from "+Tabela_String+
```

```
                                                " where login clike
'"+Login_Digitado_String+
                                                "' and password
='"+Senha_Digitada_String+"'";

                                // Executa o comando e coleta os campos
                                MsqlResult result = msql.Query(Msql_Comando);

                                while((row = result.FetchRow()) != null) {

                                        // Relata que foi tudo bem
                                        Encontrou_Registro_Boolean = true;

                                        // Coleta os dados do usuario
                                        Guarda_Login_Codigo_Usuario = row[0];
                                        Guarda_Nome_Usuario = row[1];

                                        if (Tabela_String.equals("aluno"))
                                                Guarda_Nivel_Usuario = "Aluno";

                                        if (Tabela_String.equals("professor")){

                                                Guarda_Login_Codigo_Usuario =
row[2];

                                                // Verifica se eh o gerente
                                                if
(Guarda_Login_Codigo_Usuario.equals("0"))

                                                        Guarda_Nivel_Usuario =
"Gerente";
                                                else
                                                        Guarda_Nivel_Usuario =
"Professor";
                                        }

                                }

                                // fecha conexao
                                msql.Close();

                        } catch (MsqlException e){

                             // Informa ao usuario
                             Linha_Msg.setText("mSQL Erro: "+e);
                             Beep_Toolkit.beep();
                             System.out.println("Porteiro: "+e);

                        }
                }

                // Usuario Anonimo
                if (Tipo_de_Login_String.equals("Usuario_Anonimo")){

                        // Guarda o nivel do usuario
                        Guarda_Nivel_Usuario = "Anonimo";

                        return(true);

                }

                if (Encontrou_Registro_Boolean){
```

```
                        // Guarda o nivel do usuario
                        if (Tabela_String.equals("aluno"))
                                Guarda_Nivel_Usuario = "Aluno";
                        else
                                Guarda_Nivel_Usuario = "Professor";

                        return(true);
                }

                // Sai do metodo
                return(false);
        }

        // Para montar a terceira tela
        public void Monta_Menu_Entrada_Tela(){

                // Desabilita os botoes anteriores
                Botao_Cancela.setVisible(false);
                Botao_OK.setVisible(false);
                Botao_Login_Anonimo.setVisible(false);

                Login_Aluno_TextField.setVisible(false);
                Senha_Aluno_TextField.setVisible(false);

                Imagem_Login.setVisible(false);

                Linha_Msg.setText("Contactando o sistema...");

                // Adiciona os novos botoes
                Botao_Tutorial.setVisible(true);
                Botao_Cadastro.setVisible(true);

                Imagem_Menu_Entrada.setVisible(true);

                if (!Guarda_Nome_Usuario.equals(""))
                        Linha_Msg.setText("Elabore a sua escolha
"+Guarda_Nome_Usuario+".");

        }
}
```

Código JAVA (Deamon)

```
/* Nome do Programa : Mhitsd2.java
*   Versao: 2.0
*   SO desta versao: Windows.
*   Obs: Versao para rodar com JDK 1.1.8 ou superior

*   Tipo : Aplicacao nao Applet
          (Daemon comunicador com um sistema Prolog via arquivo texto)
*   Funcao: Estabelecer uma comunicacao com uma aplicacao Prolog
           usando arquivos textos.
*   Uso: Olhe o arquivo d2.bat

*   Autor:          Adriano de O. Caminha       (caminha@dsc.ufpb.br)
```

```
*   Colaborador:    Denis Augusto A. de Souza   (denis_souza@yahoo.com)

*   Data: 13/02/2000       Hora: 10:20am

*   Bibliotecas de Classes Usadas:

        java.io.*               Conjunto de classes para controlar I/O
        java.net.*              Conjunto de classes para o trabalho com a rede
        Msql*                   Classes responsaveis pela comunicacao com uma
base de dados mSQL
        Bib_MHITS               Biblioteca do sistema MHITS
        java.util.Random        Gera os codigos dos arquivos
        java.util.Date          Pega a data e hora do sistema

*   Agradecimentos:
       Suleimam "Sam" Lalani e Kris Jamsa,
       "JAVA Biblioteca do Programador", MAKRON Books, 1997
*/

// Classes usadas
import java.io.*;
import java.net.*;
import java.util.Random;
import java.util.Date;
import Bib_MHITS;

// Para trabalhar com mSQL DataBases
import MsqlException;
import MsqlRow;
import MsqlFieldDesc;
import MsqlData;
import MsqlResult;
import Msql;

public class Mhitsd2{

        // ************** Objetos Locais **********************

        static int Porta_Int = 4000;

        // Local onde ficarao os arquivos (default)
        static String Path_Dir_String  = "c:\\tmp\\mhits\\";
        static String Query_Dir_String = Path_Dir_String+"queries";
        static String Answer_Dir_String = Path_Dir_String+"answers";

        // ************************************************

        public static void main(String args[]) throws IOException {

                // Coleta o suaurio SQL
                Bib_MHITS Biblioteca_MHITS = new Bib_MHITS();
                String Msql_Usuario = Biblioteca_MHITS.PegaUser();

                // Faz a leitura do arquivo de configuracao (mhitsd.conf)
                Faz_Leitura_Configuracao();

                ServerSocket s = new ServerSocket(Porta_Int);

                System.out.println("**********************************");
                System.out.println("*      MHITS Daemon Iniciado      *");
                System.out.println("**********************************");
                System.out.println("* Mhitsd2:  Aguardando Conexao... *");
                System.out.println("**********************************");
```

```
            try {

                    while(true){

                            // Blocks until a connection occurs:
                            Socket socket = s.accept();

                            if ( (args[1]).equals("console") )
                                    System.out.println("Cliente estabeleceu
conexao.");

                            try{

                                    // Parametros: socket, sql_host, echoing
in console
                                    new mhits_2d(socket, args[0],
args[1],Query_Dir_String,
                                                    Answer_Dir_String,
Msql_Usuario);

                            } catch(IOException e){

                                    // If it fails, close the socket,
                                    // otherwise the thread will close it:

                                    socket.close();

                            }

                     } // end while

            } finally{

                    s.close();

            }

     } // main

    /* ************************************************************
     * Metodo  para ler o arquivo de configuracao sei_govd.conf
     *       e definir o usuario sei_gov e o path dos arquivos
     *       temporarios.
     * Data: 17/04/98   Hora: 08:17am
     * Atualizacoes:
     * Parametros de Entrada: ausente
     *
     * Itens a Implementar:
     *
     * Variaveis Usadas:
     *
     * Objetos Usados:
     *
     * ************************************************************ */
     static public void Faz_Leitura_Configuracao(){

          // Define objetos/variaveis para o trabalho
             String Nome_Arquivo_Configuracao = "mhitsd.conf";
          int Dados_Lidos_Tmp;
          char Caracter_Lido_Arquivo;
          String Arquivo_Lido = new String();
```

```
            // Para retirar os comandos
            int Posicao1 = -2, Ponto1 = -2;
               int Simbolo_de_Igual1 = -2;
               String Conteudo1 = "";

               int Posicao2 = -2, Ponto2 = -2;
               int Simbolo_de_Igual2 = -2;
               String Conteudo2 = "";

               String Conteudo3 = "";
               String Conteudo4 = "";
               String Conteudo5 = "";

            try{
                  // Define novos objetos para esta tarefa
                  SequenceInputStream Sequencia_Arquivo_Lido = null;

                  FileInputStream Linha_Arquivo_Lido = new
                        FileInputStream(Nome_Arquivo_Configuracao);
                  DataInputStream Dados_Arquivo_Lido = new
                        DataInputStream(Linha_Arquivo_Lido);
                  Sequencia_Arquivo_Lido = new

      SequenceInputStream(Dados_Arquivo_Lido,Sequencia_Arquivo_Lido);

                  // Olha cada parte do arquivo gerado
                  while ((Dados_Lidos_Tmp = Sequencia_Arquivo_Lido.read() ) != -
1){

                        // Guarda o caracter lido
                              Caracter_Lido_Arquivo = (char)Dados_Lidos_Tmp;

                          // Acumula linha
                               Arquivo_Lido = Arquivo_Lido +
Caracter_Lido_Arquivo;

                  }

                  /* ***** Identifica o conteudo dos comandos **** */

                       Posicao1 = Arquivo_Lido.indexOf("Path");
                       Simbolo_de_Igual1 = Arquivo_Lido.indexOf('=',Posicao1);
                       Ponto1 = Arquivo_Lido.indexOf(';',Simbolo_de_Igual1);
                       Conteudo1 =
Arquivo_Lido.substring(Simbolo_de_Igual1+1,Ponto1);
                       Conteudo1 = Conteudo1.trim();

                       Arquivo_Lido =
Arquivo_Lido.substring(Simbolo_de_Igual1+2);
                       Posicao2 = Arquivo_Lido.indexOf("Queries");
                       Simbolo_de_Igual2 = Arquivo_Lido.indexOf('=',Posicao2);
                       Ponto2 = Arquivo_Lido.indexOf(';',Simbolo_de_Igual2);
                       Conteudo2 =
Arquivo_Lido.substring(Simbolo_de_Igual2+1,Ponto2);
                       Conteudo2 = Conteudo2.trim();

                       Arquivo_Lido =
Arquivo_Lido.substring(Simbolo_de_Igual2+2);
                       Posicao2 = Arquivo_Lido.indexOf("Answer");
                       Simbolo_de_Igual2 = Arquivo_Lido.indexOf('=',Posicao2);
                       Ponto2 = Arquivo_Lido.indexOf(';',Simbolo_de_Igual2);
                       Conteudo3 =
Arquivo_Lido.substring(Simbolo_de_Igual2+1,Ponto2);
```

```
                        Conteudo3 = Conteudo3.trim();

                        Arquivo_Lido =
Arquivo_Lido.substring(Simbolo_de_Igual2+2);
                        Posicao2 = Arquivo_Lido.indexOf("Porta");
                        Simbolo_de_Igual2 = Arquivo_Lido.indexOf('=',Posicao2);
                        Ponto2 = Arquivo_Lido.indexOf(';',Simbolo_de_Igual2);
                        Conteudo4 =
Arquivo_Lido.substring(Simbolo_de_Igual2+1,Ponto2);
                        Conteudo4 = Conteudo4.trim();

                        if (!Conteudo1.equals("")){

                                Path_Dir_String = Conteudo1;
                        }

                        if (!Conteudo2.equals("")){

                                Query_Dir_String =
Path_Dir_String+"\\"+Conteudo2;
                        }

                        if (!Conteudo3.equals("")){

                                Answer_Dir_String =
Path_Dir_String+"\\"+Conteudo3;
                        }

                        if (!Conteudo4.equals("")){

                                Porta_Int = Integer.parseInt(Conteudo4);
                        }

                        // verifica Path final
                        if (!Path_Dir_String.endsWith("\\")){

                                Query_Dir_String  = Path_Dir_String+"\\queries";
                                Answer_Dir_String = Path_Dir_String+"\\answers";

                        } else {

                                Query_Dir_String  = Path_Dir_String+"queries";
                                Answer_Dir_String = Path_Dir_String+"answers";
                        }

                        // debug
                        //System.out.println(Path_Dir_String);
                        //System.out.println(Query_Dir_String);
                        //System.out.println(Answer_Dir_String);
                        //System.out.println(Porta_Int);

                    // Fecha o arquivo...
                    Dados_Arquivo_Lido.close();

            } catch (IOException ErroIO_Observado){

                        // Informa ao usuario na linha de comando
                        System.out.println("Erro!!! (mhitsd.conf)");
                        System.out.println(ErroIO_Observado.toString());

             }

      } // end Faz_Leitura
```

```
} // end class Mhitsd2

// Classe auxiliar
class mhits_2d extends Thread {

        // *************** Objetos Locais **********************

        private Socket socket;
        private BufferedReader in;
        private PrintWriter out;

        private boolean Ativa_Console = false;

        // Objeto para enviar o conteudo do query.
        String Sistema_Prolog_Resposta = new String();

        // ****** mSQL
        // Objeto para a comunicacao com o daemon SQL
      Msql msql;
      MsqlResult result;

        String SQL_HostName = new String();
        String Msql_Usuario = new String();
        String Comando_SQL = new String();

        // Local onde ficarao os arquivos (default)
        String Path_Dir_String  = "c:\\tmp\\mhits\\";
        String Query_Dir_String = Path_Dir_String+"queries";
        String Answer_Dir_String = Path_Dir_String+"answers";

        // Para gerar o codigo
        int Pega_Primeira_Parte = 0;
        int Pega_Segunda_Parte = 0;
        String Codigo = new String();
        Random Gera_Codigo = new Random();

        // Objetos relacionados aos arquivos usados
        File Arquivo_Tmp;
        String Nome_Arquivo_Tmp = new String();

        // Contrutor da Classe
        public mhits_2d(Socket s, String Sql_Host, String Ativa_Console_String,
                        String Diretorio_Query, String Diretorio_Answer,
                        String Usuario_SQL_Usado) throws IOException{

                socket = s;
                in = new BufferedReader(new InputStreamReader(
                         socket.getInputStream()));

                // Enable auto-flush:
                out = new PrintWriter(new BufferedWriter(
                          new OutputStreamWriter(
                          socket.getOutputStream())), true);

                // Inicializa
                Query_Dir_String = Diretorio_Query;
                Answer_Dir_String = Diretorio_Answer;
                Msql_Usuario = Usuario_SQL_Usado;

                if (Ativa_Console_String.equals("console"))
                        Ativa_Console = true;
```

```
                start(); // Calls run()
        }

        // Metodo para ativar os thends
        public void run(){

                // debug
                System.out.println("Run iniciando...");

                try{
                        while (true){

                                String Mensagem = in.readLine();

                                if (Ativa_Console)
                                        System.out.println("Echoing: "+
Mensagem);

                                // ***** Comunicacao com o cliente *****
                                // Avalia Msgs do cliente
                                if (Mensagem.equals("Mata_Servidor_Super012")){

                                        out.println("END");     // Envia Msg
                                        Runtime.getRuntime().exit(0);
                                }

                                if (Mensagem.startsWith("Recebe")){

                                        // ******* Gera o arquivo com os queries
*******
                                        boolean Resposta =
Gera_Arquivo(Mensagem,Query_Dir_String);

                                        if (!Resposta)
                                                out.println("Erro: Problemas com
o arquivo de queries.");
                                }

                                if (Mensagem.equals("END")){

                                        out.println("END");          // Envia Msg
                                        break;

                                }

                        }

                        System.out.println("closing...");

                } catch (IOException e){

                } finally {

                          try{
                                socket.close();

                          }catch(IOException e) {}
                }

        } // end run

        // Metodo para coletar a resposta de um query e gravar em um arquivo
```

```
        public boolean Gera_Arquivo(String Informacao_String, String
Diretorio_Queries){

                // Gera o codigo para o arquivo
                Pega_Primeira_Parte = (Gera_Codigo.nextInt() & 255);
                Pega_Segunda_Parte = (Gera_Codigo.nextInt() & 255);
                Codigo = ""+Pega_Primeira_Parte+Pega_Segunda_Parte;

                boolean Arquivo_Gerado = false;

                // Define qual arquivo a trabalhar
                Nome_Arquivo_Tmp = Diretorio_Queries+"\\answer_"+Codigo;

                // Arquivo de saida temporario
                try{
                        File arquivo01 = new File(Nome_Arquivo_Tmp);
                        FileOutputStream Saida_Tmp = new
FileOutputStream(arquivo01);
                        PrintStream saida1 = new PrintStream(Saida_Tmp);

                        saida1.println(Informacao_String);

                        // Envia os dados da memoria para o arquivo
                        saida1.close();

                        Arquivo_Gerado = true;

                } catch (IOException e){

                        System.out.println(e);
                }

                if (Arquivo_Gerado){

                        if (Ativa_Console)
                                System.out.print("Arquivo de queries gerado ");

                        // ******* Aguarda a resposta da aplicacao Prolog
********
                        // le arquivo gerado
                        Sistema_Prolog_Resposta = Procura_Answer_Arquivo(Codigo);

                        if (Sistema_Prolog_Resposta.startsWith("Prolog")){

                                // Envia o conteudo
                                out.println(Sistema_Prolog_Resposta);
                        }

                        return(true);
                }

                // retorna o valor default caso hava problemas
                return(false);
        }


        // Procura arquivo de resposta no diretorio do sistema
        public String Procura_Answer_Arquivo(String Codigo){

                // Define qual arquivo a trabalhar
                Nome_Arquivo_Tmp = Answer_Dir_String+"\\answer_"+Codigo;
                Arquivo_Tmp = new File(Nome_Arquivo_Tmp);
                String Sistema_Prolog_Resposta = new String();
```

```
                if (Ativa_Console){

                        System.out.println(Nome_Arquivo_Tmp);
                        System.out.println("Aguardando resposta...");
                }

                while( !(Arquivo_Tmp.exists()) ){}

                //System.out.println("(Debug)"+Arquivo_Tmp.exists());

                // Le o arquivo gerado pelo sist. prolog
                Sistema_Prolog_Resposta = Faz_Leitura_Arquivo(Nome_Arquivo_Tmp);

                return(Sistema_Prolog_Resposta);
        }

        // faz a leitura de um arquivo de resposta
        public String Faz_Leitura_Arquivo(String Nome_do_Arquivo_Answer){

                // ********** Objetos Locais ************
                String Resposta_String = ""; // Zera resposta

                int Dados_Lidos_Tmp;
                char Caracter_Lido_Arquivo;

                File Arquivo_Answer = new File(Nome_do_Arquivo_Answer);

                if (Arquivo_Answer.exists()){

                   try{
                    // Define novos objetos para esta tarefa
                        SequenceInputStream Sequencia_Arquivo_Lido2 = null;

                        FileInputStream Linha_Arquivo_Lido2 = new
                                FileInputStream(Nome_do_Arquivo_Answer);

                        DataInputStream Dados_Arquivo_Lido2 = new
                                DataInputStream(Linha_Arquivo_Lido2);

                        Sequencia_Arquivo_Lido2 = new

SequenceInputStream(Dados_Arquivo_Lido2,Sequencia_Arquivo_Lido2);

                        // Le o arquivo de resposta
                        while ((Dados_Lidos_Tmp = Sequencia_Arquivo_Lido2.read()
) != -1){

                                // Guarda o caracter lido
                                Caracter_Lido_Arquivo = (char)Dados_Lidos_Tmp;

                                // Acumula linha
                                Resposta_String = Resposta_String +
Caracter_Lido_Arquivo;
                        }

                    // Fecha o arquivo...
                        Dados_Arquivo_Lido2.close();

                   } catch (IOException ErroIO_Observado){

                        // Informa ao usuario na linha de comando
```

```
                        System.out.println("Erro de leitura no arquivo do
cabe†alho padrÆo.");
                        System.out.println(ErroIO_Observado.toString());

                    }

                    // remove o arquivo
                    Arquivo_Answer.delete();

                } // if

                //System.out.println("(debug) Prolog:"+Resposta_String);

                // retorna o conteudo
                return("Prolog:"+Resposta_String);
        }

} // classe mhits_2d
```