UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

EXTRAÇÃO DE CONHECIMENTO EM BASES DE DADOS USANDO SOFTWARE LIVRE: APLICAÇÃO À ÁREA DE JOGOS ON-LINE MULTI-JOGADOR

TRABALHO DE GRADUAÇÃO

Jeferson Ornelas Wendt

Santa Maria, RS, Brasil

2007

EXTRAÇÃO DE CONHECIMENTO EM BASES DE DADOS USANDO SOFTWARE LIVRE: APLICAÇÃO À ÁREA DE JOGOS ON-LINE MULTI-JOGADOR

por

Jeferson Ornelas Wendt

Trabalho de Graduação apresentado ao Curso de Ciência da Computação
da Universidade Federal de Santa Maria (UFSM, RS), como requisito
parcial para a obtenção do grau de
Bacharel em Ciência da Computação

Orientador: Profª Drª Andrea Schwertner Charão

Trabalho de Graduação N. 226

Santa Maria, RS, Brasil

2007

Universidade Federal de Santa Maria
Centro de Tecnologia
Curso de Ciência da Computação

A Comissão Examinadora, abaixo assinada,
aprova o Trabalho de Graduação

EXTRAÇÃO DE CONHECIMENTO EM BASES DE DADOS USANDO SOFTWARE LIVRE: APLICAÇÃO À ÁREA DE JOGOS ON-LINE MULTI-JOGADOR

elaborado por
Jeferson Ornelas Wendt

como requisito parcial para obtenção do grau de
Bacharel em Ciência da Computação

COMISSÃO EXAMINADORA:

Profª Drª Andrea Schwertner Charão
(Presidente/Orientador)

Prof. MSc. João Carlos Damasceno Lima (UFSM)

Prof. Dr. Cesar Tadeu Pozzer (UFSM)

Santa Maria, 02 de março de 2007.

RESUMO

Trabalho de Graduação
Curso de Ciência da Computação
Universidade Federal de Santa Maria

EXTRAÇÃO DE CONHECIMENTO EM BASES DE DADOS USANDO SOFTWARE LIVRE: APLICAÇÃO À ÁREA DE JOGOS ON-LINE MULTI-JOGADOR

Autor: Jeferson Ornelas Wendt
Orientador: Profª Drª Andrea Schwertner Charão
Local e data da defesa: Santa Maria, 02 de março de 2007.

Este trabalho trata da aplicação de técnicas de extração de conhecimento na base de dados do jogo Ryudragon, que é um jogo da categoria *Massively Multiplayer On-line Game* baseado em turnos. Através destas técnicas, espera-se levantar padrões e associações de valores que auxiliem a equipe de desenvolvimento a aprimorar o sistema Ryudragon para seus usuários. Para alcançar estes objetivos, utiliza-se um Software Livre denominado Yale, que reúne diversos recursos de suporte à mineração de dados. Com base nesta ferramenta, desenvolveu-se um sistema denominado Ikasu, que visa facilitar a seleção de técnicas e a apresentação dos resultados para a aplicação em questão.

Palavras-chave: Mineração de dados; *data marts*; aplicações para KDD; yale; MMO; jogo online; Ryudragon.

ABSTRACT

Trabalho de Graduação
Graduation in Computer Science
Universidade Federal de Santa Maria

KNOWLEDGE EXTRACTION FROM DATABASES USING FREE SOFTWARE: AN APPLICATION TO ONLINE MULTI-PLAYER GAMES
Author: Jeferson Ornelas Wendt
Advisor: Prof[a] Dr[a] Andrea Schwertner Charão

The main objective of this work is to apply Knowledge Extraction of Data technics in database of Ryudragon turn-based game. We wait in this project, to find patterns and association rules of values which helps to improve the system of Ryudragon to its users. To achieve this objectives, we will use the free software called Yale, which have recourses of support of Data Mining. With this tool as a base, we will develop the system named Ikasu, which will facilitate the selection of technics and the presentation of the results of the Yale application.

Keywords: data mining; data marts; KDD applications; Yale; MMO; online games; Ryudragon.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A indústria de jogos para computadores cresceu de forma significativa nos últimos anos (GAMESPOT, 2006; WAN; XU; ZHOU, 2006), introduzindo novas exigências as empresas que ingressam neste competitivo mercado. Como parte deste cenário, tem-se os jogos designados pelo termo *Massively Multiplayer On-line Game* (MMOG ou MMO) – Jogo *On-line* Maciçamente Multi-jogador – que se caracterizam por envolver muitas pessoas jogando ao mesmo tempo e interagindo em um mesmo ambiente computacional (WAN; XU; ZHOU, 2006). Neste tipo de *software* a satisfação pessoal do usuário é essencial para a sobrevivência do jogo (TATSUMOTO; THAWONMAS, 2004), pois sem uma comunidade ativa dentro do sistema, o jogador não teria como se divertir de forma mais efetiva e deixaria de usar o ambiente.

A empresa Decadium Studios Ltda., situada na Incubadora Tecnológica da Universidade Federal de Santa Maria, atua no segmento descrito acima através de um MMO denominado Ryudragon. Buscando se adequar a padrões corporativos mais avançados, a empresa deseja pesquisar formas de aumentar a satisfação de seus clientes jogadores. O presente trabalho representa um primeiro passo neste sentido, que consiste na análise dos dados do Ryudragon a fim de obter subsídios para futuras melhorias no ambiente de jogo. O método escolhido para se alcançar este objetivo foi a mineração de dados (*data mining*), também conhecida como Descoberta de Conhecimento em Banco de Dados (*Knowledge Discovery in Databases* – KDD).

A mineração de dados é uma sub-área da KDD, que é responsável pela procura por padrões, análise de variáveis, aplicação de regras associativas e métodos estatísticos sobre um grande conjunto de informações (LAROSE, 2005; JúNIOR, 2003). Segundo Larose (LAROSE, 2005) e Hand (HAND; MANNILA; SMYTH, 2001), a mineração de dados é a tecnologia que melhor se aplica à descoberta de conhecimento em bases de da-

dos de grande porte e serve para encontrar relacionamentos inesperados e interpretações dos dados que sejam úteis ao seu proprietário.

O sistema Ryudragon possui cerca de doze mil e oitocentos jogadores cadastrados (dados de dezembro de 2006) com informações espalhadas por cerca de noventa tabelas, algumas sendo continuamente atualizadas a cada vinte minutos. Devido à quantidade de dados a serem analisados nas tabelas do Ryudragon (cerca de 900 campos para o atual número de usuários), a formação de relacionamentos entre as informações e interpretações do conhecimento que pode ser retirado dos dados se torna difícil sem a ajuda de alguma técnica ou ferramenta de suporte como a mineração de dados. As técnicas de mineração de dados aplicadas diretamente nas bases de dados do jogo podem trazer informações antes não percebidas pela equipe de desenvolvimento e manutenção do Ryudragon, como padrões de comportamento numérico do sistema e padrões de comportamento dos jogadores (KENNERLY, 2003). Com base neste tipo de informações, decisões mais embasadas podem ser tomadas para se inserir ou efetuar modificações nas características do sistema.

Para realizar esta tarefa é necessário um sistema que explore algoritmos de mineração de dados de forma particular às necessidades do Ryudragon. A ferramenta escolhida para auxiliar nesta tarefa foi Yale (MIERSWA et al., 2006), que é uma ferramenta de código aberto distribuída sob uma licença de Software Livre. Esta ferramenta é modular, programada em Java, possui uma interface gráfica, uma boa documentação e um suporte por parte dos desenvolvedores e, principalmente, funciona como uma biblioteca de funções, sendo que seus módulos podem ser inseridos em uma aplicação personalizada para o sistema do Ryudragon.

A fim de atender às necessidades da equipe do Ryudragon, nasceu a idéia da criação de uma interface remota para o sistema. O Ryudragon é um jogo que funciona ininterruptamente durante dois meses, pára e depois recomeça. O sistema para mineração de dados planejado para o jogo deverá estar acessível durante este período de funcionamento e não deverá ser executado em máquinas externas ao ambiente da Decadium. Uma solução para que este sistema seja acessível de qualquer lugar a qualquer hora é a disponibilização de uma interface Web personalizada para o sistema de mineração de dados do Ryudragon que será executado nos servidores da Decadium. Deste apanhado de necessidades surgiu a idéia da construção de uma ferramenta para realização de experimentos de mineração

de dados, adaptada as necessidades do Ryudragon e que possa ser acessada remotamente via *Web*. Esta ferramenta, denominada Ikasu, constitui a principal contribuição deste Trabalho de Graduação.

No próximo capítulo, serão abordados os aspectos teóricos que embasaram este trabalho, incluindo conceitos sobre mineração de dados, armazém de dados, a metodologia para mineração de dados CRISP-DM, utilizada neste trabalho, além de uma introdução aos elementos de MMO relevantes no contexto em questão. Além disto, serão apresentadas nesta parte algumas das ferramentas de Software Livre para KDD, incluindo o Yale. Após, no capítulo 3, será apresentado o desenvolvimento da ferramenta Ikasu, iniciando pelos aspectos relevantes do Ryudragon, seguidos por uma visão geral da ferramenta, incluindo seus aspectos de funcionamento, modelagem e questões relacionadas à sua implementação. No capítulo 4, apresenta-se experimentos e alguns resultados alcançados até o momento referentes a mineração de dados nas bases do Ryudragon. No capítulo final apresenta-se a conclusão do trabalho, apresentando seu estado atual face ao planejamento inicial e atividades a serem realizadas no futuro.

2 JOGOS MMO E DESCOBERTA DE CONHECIMENTO

Este capítulo está dividido em duas partes: a dos conceitos por trás do presente trabalho e a das ferramentas de Software Livre usadas para análise e mineração de dados.

Na primeira parte temos quatro subdivisões. Na primeira subseção, será apresentada uma breve introdução a tipos de algoritmos de KDD, mais especificamente, na área de mineração de dados e sua diferença com a área de consultas de apoio a decisão (OLAP). Após, serão explicados alguns conceitos sobre armazéns de dados e *Data Marts*. Na terceira subdivisão, será apresentada a metodologia utilizada para experimentos de mineração de dados (CRISP-DM). Na subdivisão final da parte de conceitos teremos uma introdução sobre jogos MMO.

Na segunda parte do capítulo, serão apresentados primeiramente os detalhes da ferramenta para mineração de dados Yale e, na divisão seguinte, serão apresentadas outras ferramentas de Software Livre usadas em KDD.

2.1 Conceitos

2.1.1 Descoberta de conhecimento em banco de dados

A área de Descoberta de conhecimento em Dados (KDD) é empregada na busca de informações relevantes sobre um conjunto de dados. De modo mais específico, as aplicações de mineração de dados são de grande valia para as empresas que possuem grandes bases de dados (BERRY; LINOFF, 2004). A mineração de dados (*Data Mining*) é responsável pela procura por padrões, análise de variáveis, aplicação de regras associativas e métodos estatísticos sobre um grande conjunto de informações (LAROSE, 2005). Ela pode ser classificada, segundo Larose, em seis categorias a saber: tarefas de descrição, tarefas de estimativa, tarefas de predição, de classificação, de aglomeração e de associação (LAROSE, 2005).

As tarefas de **descrição**, ou *profiling* englobam a análise da base de dados em busca de formas de explicar padrões simples e comportamentos numéricos nos dados (BERRY; LINOFF, 2004). Segundo Larose, estas tarefas usam algoritmos que produzem saídas intuitivas a não-técnicos em mineração de dados, como as árvores de decisão (BREIMAN et al., 1984). O objetivo do modelo de descrição é descrever todos os dados ou o processo que os gerou (HAND; MANNILA; SMYTH, 2001). As tarefas de descrição se beneficiam da análise exploratória dos dados (*Exploratory Data Analysis* – EDA), que consiste na procura por padrões usando ferramentas gráficas para mostrar os conjuntos de dados de forma que permitam uma análise intuitiva por parte do usuário (HAND; MANNILA; SMYTH, 2001).

As tarefas de **estimativa**, segundo Berry e Linoff (BERRY; LINOFF, 2004), são as tarefas que lidam com uma entrada com várias variáveis, e estimam uma saída classificada. Ou seja, segundo Larose (LAROSE, 2005), as tarefas de estimativa assimilam uma entrada numérica e usam valores históricos dos resultados para o conjunto de entrada para estimar o resultado para novas entradas.

As tarefas de **predição** são similares às de classificação e estimativa, estando os resultados projetados no futuro (LAROSE, 2005) (BERRY; LINOFF, 2004). A principal diferença que separa a predição das outras técnicas, segundo Berry e Linoff (BERRY; LINOFF, 2004), é que alguns dos dados de entrada não são conhecidos e o resultado final dependerá de quão bem essas variáveis desconhecidas foram introduzidas no modelo. O que define a diferença chave entre tarefas de descrição e predição, segundo Hand (HAND; MANNILA; SMYTH, 2001), é que na primeira se analisa uma gama variável de dados para dar um resultado semântico à visualização destes dados. Já na segunda, o objetivo é a predição de uma única variável resultado. As técnicas de mineração mais comumente utilizadas para os modelos de predição, segundo Larose, são as redes neurais (REED; II, 1999), as árvores de decisão (BREIMAN et al., 1984) e os métodos da k-vizinhanças mais próximas (HAND; MANNILA; SMYTH, 2001).

As tarefas de **classificação**, segundo Chen e Han (CHEN; HAN, 1996), dispõem-se a classificar um conjunto de entrada baseado nos valores de uma ou mais variáveis componentes deste grupo. Hand (HAND; MANNILA; SMYTH, 2001) considera a classificação como uma variante das tarefas de predição em que a variável a ser predita é categórica, ou seja, resulta em uma classificação de categoria. Segundo Larose, a maioria dos métodos

usados para tarefas de predição podem ser usados nas tarefas de classificação.

A **clusterização** (ou segmentação (CHEN; HAN, 1996)) é a tarefa de segmentar um grupo heterogêneo em subgrupos com características homogêneas, chamados *clusters* (LAROSE, 2005) (BERRY; LINOFF, 2004). Segundo Larose, a clusterização difere das tarefas de classificação por não ter uma variável alvo para separar a entrada em grupos distintos. Na clusterização o algoritmo procura agrupar as entradas em subgrupos que possuem certas características similares entre si, mas diferem do resto dos *clusters*. Algumas técnicas usadas para clusterização são a clusterização hierárquica, *k-means* e redes Kohonen (KOHONEN, 1982).

Por fim, as tarefas de **associação** preocupam-se em analisar relações específicas de atributos dentro do universo de dados. Também são conhecidas como análises por afinidade e análises de cestas de mercado (*market basket analysis*) (BERRY; LINOFF, 2004). São usadas para descobrir regras para quantificar o relacionamento entre dois ou mais atributos.

Existe uma classe de aplicativos e técnicas que se assemelham aos de mineração de dados, que são os aplicativos OLAP (MOLINA; ULLMAN; WINDOM, 2001). Estas técnicas consistem em consultas complexas feitas diretamente as bases de dados com o intuito de pesquisar informações relevantes ao modelo de negócios. Segundo Molina, as consultas de mineração de dados podem ser consideradas uma extensão do modelo OLAP, sendo que a diferença principal, informalmente, está no modo como a abordagem do problema é feita. Ainda segundo Molina, no modelo OLAP, os analistas já sabem exatamente o foco que necessitam procurar nas consultas e no modelo de mineração de dados, o sistema é levado a apontar possíveis focos para a análise.

2.1.2 Armazéns de dados

Na área de descoberta de conhecimento em base de dados, existem ferramentas e técnicas específicas para se lidar com os dados a serem analisados e um ponto deve ser particularmente bem resolvido: onde serão feitas estas análises. Para proporcionar uma base onde dados de várias fontes dentro de um modelo de negócios sejam armazenados de forma padronizada e bem indexada, surgiram os armazéns de dados (BERRY; LINOFF, 2004).

Armazéns de dados são tabelas formadas pela captação periódica de dados de várias

outras tabelas correntes dentro da empresa, que são gravados de forma padronizada voltada à futura consulta por analistas, formando uma memória empresarial (BERRY; LINOFF, 2004; BONIFATI et al., 2001). O uso deste tipo de técnica se dá porque as tabelas usadas no dia a dia dos sistemas corporativos geralmente não são projetadas para suportar a carga de entrada e saída geralmente exigida por algoritmos de mineração de dados e os armazéns são projetados exclusivamente para o suporte a decisão (BERRY; LINOFF, 2004). Outros motivos que podem ser citados são que os dados das tabelas correntes ou a tecnologia de suporte podem ser fisicamente diferentes dos necessitados pelos processos analíticos e a forma de apresentação dos dados nas tabelas correntes podem dificultar as análises (IMMON, 2002). As tabelas correntes em uma empresa são denominadas base operacional (IMMON, 2002; BERRY; LINOFF, 2004; KIMBALL; ROSS, 2002) e, para que a base de análise seja construída, costuma-se usar uma classe de ferramentas especiais conhecidas como ferramentas ETL (IBM, 2004; JUKIC, 2006; KIMBALL; ROSS, 2002) (sigla em inglês para Ferramentas de Extração, Transformação e Carregamento).

O processo de ETL é dividido nas fases de Extração, Transformação e Carregamento (KIMBALL; ROSS, 2002). Na fase de extração, os dados a serem usados na base final são procurados e extraídos da base operacional. Na fase de transformação estes dados são corrigidos usando-se conversão de valores, normalização, e filtragens para serem gravados no formato correto na futura base. A fase de carregamento se propõe a gravar de forma correta os dados na base final de pesquisas. A modelagem das ferramentas ELT geralmente é a fase que toma mais tempo dentro de um processo de modelagem de um armazém de dados (MOSS; ATRE, 2003; PONNIAH, 2001).

Seguindo o conceito de armazéns de dados, temos uma extensão de suas funcionalidades voltada a pontos específicos do modelo de negócios, os *Data Marts* (BONIFATI et al., 2001). *Data Marts* são tabelas menores que usam os mesmos princípios dos armazéns de dados mas têm um escopo mais limitado (JUKIC, 2006). Existem três abordagens importantes para a construção de *Data Marts*. Os *Data Marts* podem ser construídos a partir de uma tabela de armazém de dados centralizada que fornece todas as informações necessárias conforme a modelagem proposta por Inmom (IMMON, 2002). Outra forma, proposta por Kimball (KIMBALL, 1998), vê os *Data Marts* como parte da modelagem do armazém de dados, que é formado por grupos de *Data Marts*. A terceira discute a produção de *Data Marts* separados usando ferramentas ELT distintas para cada um. Se-

gundo Jukic (JUKIC, 2006), o custo de implementação das ferramentas ELT para cada *Data Mart* e a falta de conectividade entre eles deixa esta como sendo a pior opção de modelagem.

2.1.3 Metodologia CRISP-DM

Em busca de um processo padrão para a modelagem de soluções para problemas de mineração de dados, as empresas DaimlerChrysler, SPSS e NCR reuniram esforços para a criação do CRISP-DM (PETE CHAPMAN , NCR). O CRISP-DM (*Cross-Industry Standard Process for Data Mining*, ou Processo Padrão Inter-industrial para Mineração de Dados, consiste em um processo padrão composto de seis fases para a modelagem gradual de problemas de mineração de dados para qualquer tipo de negócio ou empresa. Estas seis fases representam o ciclo de vida de um projeto de mineração de dados e são implementadas de forma adaptativa (LAROSE, 2005).

Cada fase do processo pode ser repetida caso uma adaptação em uma fase mais avançada seja necessária para que o modelo de mineração seja melhorado (LAROSE, 2005). Cada uma das seis fases é responsável por uma tarefa particular identificada pela equipe de desenvolvimento do padrão CRISP-DM como visto a seguir.

A primeira fase, a fase de compreensão do negócio, consiste em uma análise por parte da equipe responsável pela mineração de dados do negócio como um todo. Nesta fase, deve-se definir os objetivos do projeto nos termos do sistema analisado, tentar passar estes objetivos e as possíveis dificuldades para uma definição de problema de mineração de dados e esboçar estratégias para alcançar estes objetivos.

Na segunda fase, a de compreensão dos dados, deve-se dar atenção às formas de coletar os dados, fazer análises exploratórias nas bases para melhor familiarização e selecionar, se possível, partes de dados que apresentem padrões relevantes e avaliar a sua qualidade quanto a consistência.

Na terceira fase do processo, deve-se preparar os dados para o uso nas fases subseqüentes. Aqui seleciona-se as variáveis que possivelmente serão úteis para a análise, transforma-se ou mescla-se variáveis em informações úteis e retira-se informações incorretas ou completa-se campos faltantes na base.

Na quarta fase, a modelagem, deve-se escolher e aplicar técnicas de mineração de dados nos dados preparados. Aqui, as particularidades dos algoritmos escolhidos podem

fazer com que se deva voltar às outras fases para se adaptar algum dado faltante para sua execução (LAROSE, 2005).

Na quinta fase, a de avaliação do modelo, estuda-se se o algoritmo utilizado alcançou os objetivos previstos na primeira fase e se deveria cobrir outra faceta do negócio.

Na última fase, publicam-se os resultados alcançados pelo modelo em formatos inteligíveis para os analistas.

2.1.4 Jogos MMO

Os jogos do tipo *Massively Multi-player Online* (MMO), termo criado pela empresa Eletronic Arts, podem ser definidos segundo Bosser (BOSSER, 2004) como uma aplicação que tem a propriedade de manter centenas de jogadores interagindo entre si dentro de um ambiente virtual. A maioria são jogos que possuem um universo persistente (WAN; XU; ZHOU, 2006) em que os usuários possuem uma representação dentro do jogo chamada **avatar**. Quando saem do sistema, o estado do avatar é mantido no servidor até que o jogador volte ou alguma regra do sistema ou evento externo modifique-o.

Nos MMO's, a noção de comunidade é muito mais acirrada que em outros estilos de jogo, possuindo sistemas políticos, culturais e econômicos embasados pela participação dos usuários, proporcionando assim a criação de sociedades paralelas dentro do sistema (CHICK, 2003). Ou seja, o foco principal dos MMOG's são os usuários e formas de mantê-los interessados e atuantes dentro do jogo. Neste tipo de jogo, os usuários têm uma gama de opções, ganham e perdem pontos, habilidades e recursos dentro do sistema a medida que trabalham sozinhos ou interagindo com outros jogadores (BUCHANAN; ESS, 2005).

As habilidades dos jogadores dentro do jogo, as formas como estes usuário ganham e perdem pontos ou qualquer tipo de recurso dentro do sistema podem ser denominadas funcionalidades (*features*) do jogo (KENNERLY, 2003; BOSSER, 2004).

Alguns tipos de MMO's disponibilizam ao usuário a opção de escolher entre classes de avatares (TATSUMOTO; THAWONMAS, 2004), sendo que estas classes diferem no modo com que o sistema aplica suas funcionalidades sobre o avatar do jogador. O equilíbrio geral entre estas diferenças de classes de avatares, fazendo com que uma não seja mais beneficiada que outra no contexto geral pode ser denominada de balanceamento do jogo (KENNERLY, 2003). Nenhum jogo está totalmente equilibrado (APPELCLINE,

2003), então medidas devem ser tomadas para gradualmente alcançar-se um ponto aceito pela maioria dos usuários, porque segundo Carpenter (CARPENTER, 2003), um jogo em desequilíbrio desagrada os jogadores e gera publicidade negativa para a empresa desenvolvedora. Este balanceamento pode ser alcançado através de mudanças empíricas baseadas em sugestões dos usuários ou iniciativa empírica dos desenvolvedores e também através de pesquisas nas bases de dados pela procura de padrões de comportamento dos usuários e medidas de desempenho baseadas nos números do sistema (KENNERLY, 2003). Nesta última, uma das soluções propostas por Carpenter (CARPENTER, 2003) é o uso da ferramenta de análise de risco @Risk (PALISADE, 2006) nos dados do sistema de ataque dos jogos, para a criação de modelos para análises visando o equilíbrio numérico em MMO's.

2.2 Ferramentas de Software Livre para KDD

O número de ferramentas disponíveis para mineração de dados é grande e, em sua maioria, trata-se de *software* proprietário de custo elevado (TWIKI, 2006). Ferramentas relacionadas ao *software* Clementine da SPSS custam de seiscentos a sete mil e quinhentos dólares. Outro *software*, o Knowledge Studio da Angoss, disponibiliza sua licença comercial por dez mil e quinhentos euros. O Compumine Rule Discovery System da Compumaine disponibiliza sua licença comercial por no mínimo dois mil euros. Outras ferramentas como o GeneXproTools 4.0 da gepsoft, que custa trinta e três euros, são alternativas de mais baixo custo, mas são limitadas a algoritmos específicos. Devido aos altos preços das licenças ou pelas limitações de algumas ferramentas proprietárias, optou-se por pesquisar soluções em Software Livre que apresentassem uma gama de opções para soluções de problemas de mineração de dados e que não fossem voltadas a mineração de textos, estando estas citadas nas seções seguintes.

2.2.1 Yale

A ferramenta Yale (MIERSWA et al., 2006) foi criada para ser um ambiente para experimentos na área de *machine learning* e mineração de dados (FISCHER; KLINKENBERG; MIERSWA, 2006). O Yale se propõe a fazer com que problemas de mineração de dados sejam rapidamente prototipados e testados dentro de seu ambiente. Para isso, a ferramenta oferece cerca de quatrocentos operadores entre operadores estatísticos, de

machine learnig, de pré-processamento, de transformação de espaço de características, de avaliação e otimização de experimentos e de visualização. Além dos operadores, o Yale trabalha com um conceito de árvore de operadores, onde existe um operador raiz, chamado de experimento, e os operadores colocados após ele funcionam como uma corrente de operadores, onde a saída de um corresponde a entrada de outro. A própria interface do Yale avisa na hora de execução do experimento se os operadores foram colocados na posição correta na corrente. Um exemplo de experimento e sua corrente de operadores pode ser vista na parte esquerda da imagem 2.1, que condiz com a interface gráfica do Yale.

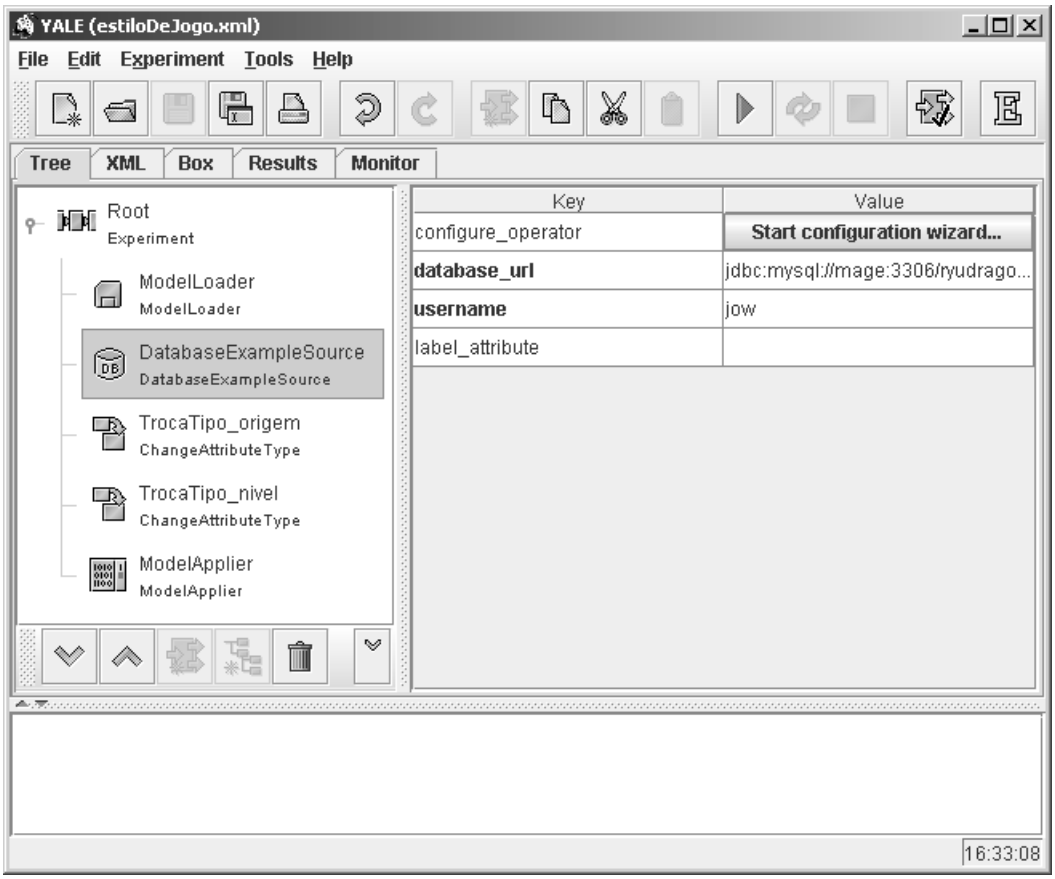

Figura 2.1: Yale - Interface Gráfica. A esquerda a árvore de operadores e à direita as configurações do mesmo.

Para se executar um experimento, cria-se uma árvore de experimentos, insere-se os operadores na ordem correta e manda-se executar. Caso se esteja usando a interface gráfica do Yale, o resultado será mostrado na tela em uma janela. Cada experimento possui um arquivo de configuração que informa as variáveis iniciadas para cada operador, bem como a ordem dos operadores na corrente. Caso se queira repetir o experimento em outra

máquina ou usá-lo em outras instâncias do Yale, basta carregar este arquivo e executar.

O Yale pode ser usado como uma ferramenta instalada no computador do usuário, ou como um *framework* para desenvolvimento de sistemas. Para mais detalhes, veja o apêndice A.

2.2.2 Outras ferramentas

Além de Yale, há outras ferramentas de Software Livre voltadas à análise de dados. Algumas delas são citadas a seguir:

- Weka (WITTEN; FRANK, 2005): consiste em uma biblioteca em Java com uma coleção de algoritmos na área de aprendizado de máquina para resolução de problemas de mineração de dados. Inclui algoritmos de pré-processamento de dados, classificação, regressão, clusterização, regras de associação e visualização. Existem vários projetos que se beneficiam com o Weka, incluindo o *framework* Yale.

- Orange (DEMSAR, 2004): ferramenta licenciada sob GPL (*General Public License*) construída para a resolução de vários tipos de tarefas de mineração de dados. Escrita em linguagem C++, necessita de Python para funcionar. Também funciona de forma modular como o Yale, pode ser acoplada a um *software* particular e seus módulos são acessados utilizando-se *scripts* Python. Os módulos de Orange incluem tarefas como entrada de dados, manipulação e pré-processamento de dados de entrada, métodos de classificação, regressão, predição, aplicação de regras associativas, clusterização e visualização.

- FastMapDb (PATERLINI et al., 2005): ferramenta desenvolvida no Brasil para o auxílio na área da análise exploratória dos dados (EDA), baseado no algoritmo FastMap (FALOUTSOS; LIN, 1995). Foi programada em C++ e usa a biblioteca gráfica OpenGL para apresentar de forma gráfica os resultados. FastMapDb mapeia os objetos de uma base de dados para uma espaço euclidiano, procurando preservar as distâncias entre os objetos e evitar distorções neste mapeamento. Após isto, disponibiliza os dados em um *display* gráfico, onde o analista pode manipulá-los usando ferramentas disponibilizadas pelo *software* para uma melhor compreensão de sua semântica.

3 A FERRAMENTA IKASU

A ferramenta Ikasu visa proporcionar aos administradores uma forma de acesso remota a experimentos estatísticos e de mineração de dados relacionados de uma forma personalizada às condições do jogo Ryudragon e sua base de dados.

Este capítulo descreve o projeto e a implementação da ferramenta Ikasu, começando por uma explanação sobre algumas particularidades do jogo Ryudragon, seguido por uma visão geral da ferramenta, um detalhamento de seu projeto e, por fim, uma discussão das principais questões relacionadas à sua implementação.

3.1 O jogo Ryudragon

O Ryudragon é um jogo MMO baseado em turnos que é jogado através de um navegador Web. Para começar a jogar, o usuário deve se cadastrar no sistema, recebendo assim uma conta, denominada **feudo**. No feudo, o jogador pode efetuar ações como administrar recursos (figura 3.1) , criar construções e exércitos, e interagir com outros jogadores. Toda a ação que o jogador deseja proceder em seu feudo é efetuada através da interface do jogo (figura 3.1) e em um período de tempo denominado **turno** (figura 3.1). Isto significa que o jogador tem este período de tempo para indicar ações para seu feudo através da Web e que, ao final deste período, estas ações são efetivadas no servidor central do jogo, começando assim um novo turno.

Os turnos do Ryudragon são de vinte minutos e o jogo conta atualmente com cerca de doze mil e oitocentos jogadores cadastrados, com uma média de cento e doze jogadores *on-line* por turno (dados de dezembro de 2006).

O jogo transcorre em um número de turnos limitados dentro de um período de dois meses chamado **era**. Neste período, o jogador deve construir exércitos dentro do sistema para atacar os outros usuários e assim ganhar pontos, denominados **XP** (pontos de exper-

Figura 3.1: Interface Ryudragon - interface de acesso ao Ryudragon

iência). O jogador que obtiver mais pontos (XP) dentro da era é o vencedor. Para efetuar um ataque, primeiramente o jogador deverá escolher um alvo em uma página com uma lista de outros feudos. Após, o jogador deverá destacar uma quantidade de soldados de seu exército enviá-los para o alvo.

O ataque demora um certo número de turnos para chegar ao destino e o resultado da batalha se dá em um cálculo entre as unidades destacadas para o ataque e as unidades defendendo o feudo atacado. Para um melhor entendimento do funcionamento do jogo, em especial conceitos que serão diretamente relacionados à aplicação das técnicas de mineração de dados, a seguir serão explanados como se agrupam as categorias de feudos, como funciona o sistema de XP e como os usuários fazem para personalizar suas estratégias usando o mecanismo destinado para este fim dentro do sistema do jogo.

No início de cada era, algumas informações sobre as contas de usuários devem ser recadastradas. Dentre estas informações, está a escolha da **origem** a qual o jogador quer pertencer (figura 3.2). O usuário pode escolher apenas uma dentre as cinco disponíveis e a manterá até o fim da era. Em nível de sistema, a origem é uma classificação da conta do usuário que influi nos resultados de suas ações. Ou seja, o sistema pode gerar um resul-

tado diferente para uma mesma ação de feudos de origens diferentes. Este sistema serve para que os jogadores possam montar estratégias condizentes a seu estilo de jogo em nível macro, pois poderão existir muitos outros feudos da mesma origem. O ponto crítico deste mecanismo é alcançar um patamar em que a origem com seu conjunto de comportamentos diferenciado seja considerada equilibrada por parte do usuário. Caso os jogadores considerem uma origem como em desvantagem as outras, não a escolherão, causando assim um comportamento indesejado pelos desenvolvedores do jogo que esperam que as escolhas de origens se dêem de forma uniforme dentro do número total de feudos.

Figura 3.2: Tela inicial do Ryudragon - raças disponíveis

Outro sistema relevante ao entendimento do Ryudragon é o dos pontos de experiência (XP). O XP de um jogador corresponde ao somatório de pontos adquiridos por ele como resultado de certas ações dentro do jogo. Cada ação que dá pontos, como os ataques, se bem sucedida, acrescentará um valor ao XP do usuário. Este número é zerado a cada começo de era e serve como medida, junto a outros fatores, como elemento de posicionamento do usuário no *ranking*, para determinar o ganhador do jogo e para determinar o **nível** do jogador. O nível corresponde a um valor de um a dez dentro de uma faixa de valores de XP. A faixa do nível 1 começa em zero até um certo valor, onde começa o nível dois e o nível dez começa no valor final do nível nove e se estende ao infinito. Este valor de nível é um sistema usado para regrar vários outros subsistemas do jogo, dentre eles, quem o jogador poderá atacar dentro do *ranking*, os tipos de tropas que poderá construir dentre outros. O ponto crítico deste mecanismo é determinar a faixa de XP para cada nível e determinar o valor dado por cada ação no jogo de forma que a média de ações efetuadas que dão pontos por nível seja considerada justa por parte do usuário, levando-se em conta as origens. Este sistema pode ser visualizado na figura 3.3

O último sistema a ser explanado para um melhor entendimento do Ryudragon é denominado **árvore tecnológica** (figura 3.4.

Figura 3.3: Ryudragon - Niveis e Barra de Experiência.

A árvore tecnológica consiste em um mecanismo para o jogador com o objetivo de granularizar as opções estratégicas dentro do jogo. É uma estrutura em grafo em que cada nodo corresponde a um tipo de diferenciação do comportamento do sistema a dada ação do usuário. Ou seja, funciona como uma personalização estratégica do feudo do usuário. A cada nível, o usuário recebe um número de pontos em que ele pode gastar em certos nodos de sua árvore, podendo assim mudar os comportamentos do sistema que mais condizem com sua conduta de jogo. A escolha dos nodos é limitada pelo nível do jogador ou o número de pontos que o mesmo tem para gastar na árvore ou por algum nodo pré-requisito, sendo que no último nível o jogador pode escolher qualquer um dos nodos se possuir pontos suficientes e tiver escolhido anteriormente todos os nodos pré-requisito para dada tecnologia. O ponto crítico deste sistema é igual ao apresentado no problema das origens, mas de uma forma mais ampla, pois as possibilidades de escolha são maiores, posto que existem atualmente vinte e seis tecnologias para cinco origens.

O Ryudragon é composto por outros sistemas além dos citados anteriormente como sistemas de espionagem, construções de edificações, construção de tropas, obtenção de recursos e muitos outros que podem ser melhor entendidos consultando-se a documentação

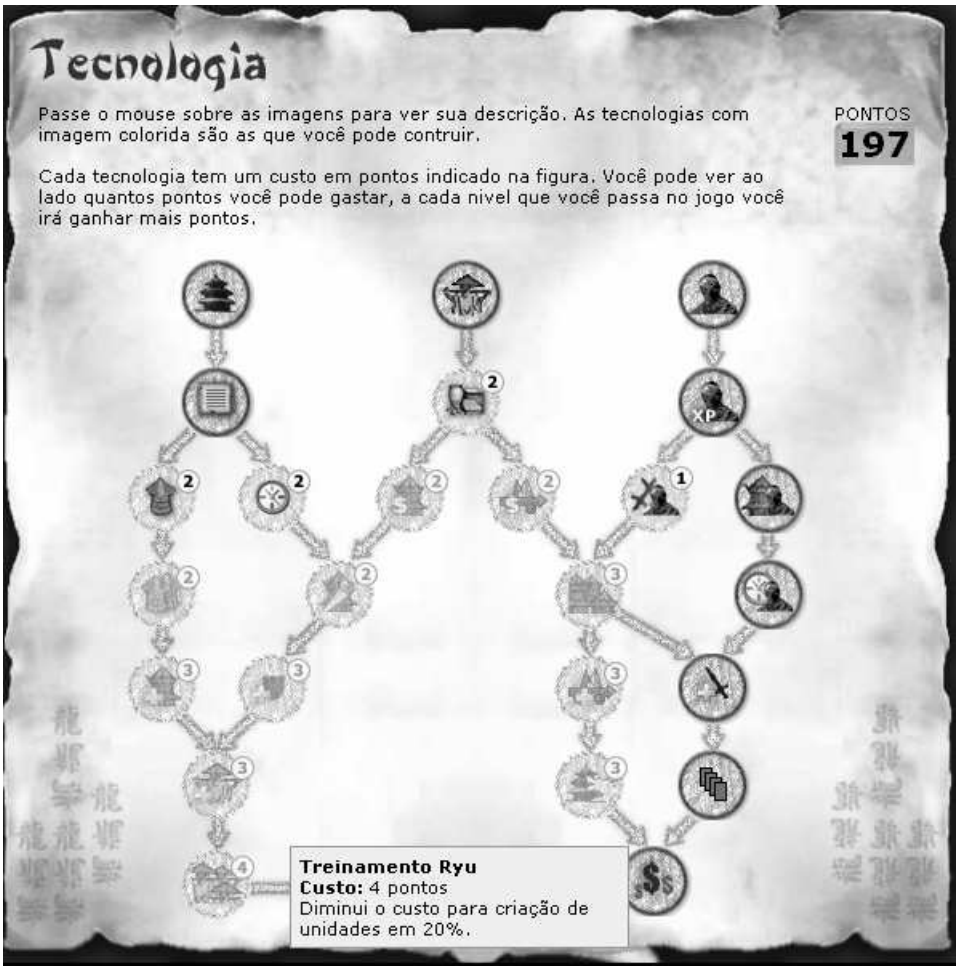

Figura 3.4: Ryudragon - Árvore Tecnológica.

do jogo (LTDA, 2005). A maioria destes sistemas, além dos já citados, tem a possibilidade de apresentar resultados relevantes para a equipe de desenvolvimento e manutenção do Ryudragon se investigados com as técnicas de mineração de dados. Estas possibilidades e a necessidade de poder acessar os experimentos de forma remota levaram ao projeto da ferramenta Ikasu, descrita na próxima seção.

3.2 Visão geral

A ferramenta Ikasu foi projetada como um recurso de apoio à decisão a ser utilizado pela equipe de manutenção do Ryudragon. Esta ferramenta capta informações de uma base principal, modifica estas informações, as grava em uma base de análises com a ajuda de tabelas auxiliares e, sobre estas informações, usa o *framework* Yale para realizar experimentos de mineração de dados. A estrutura deste sistema pode ser observada na figura 3.5.

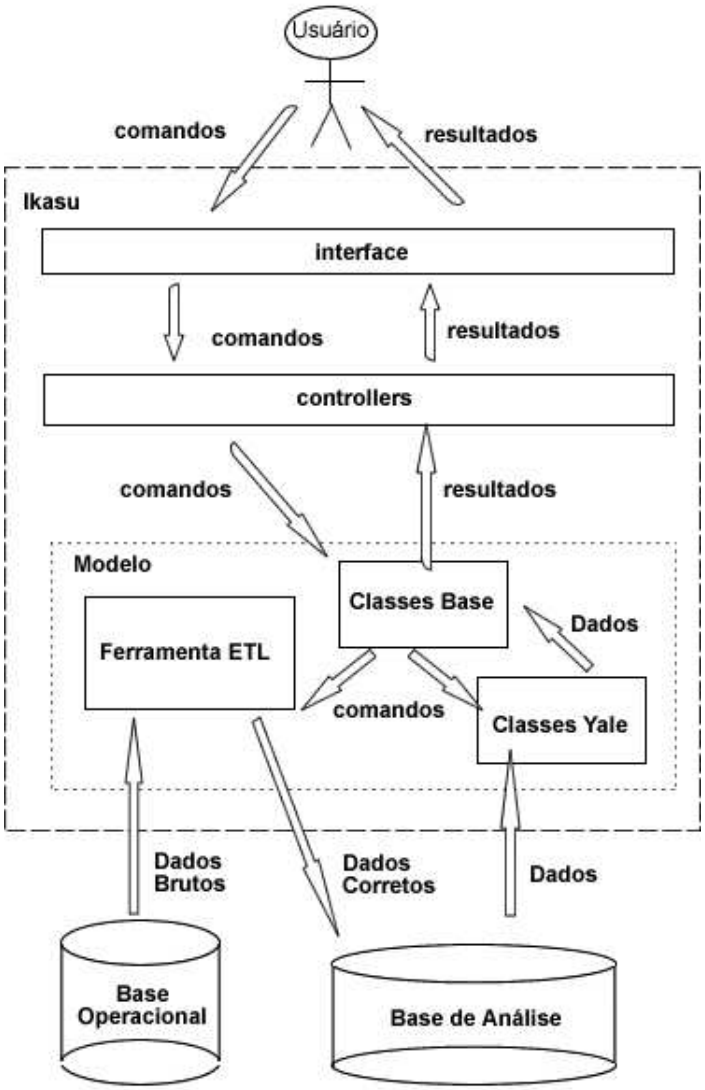

Figura 3.5: Ikasu - Diagrama geral de funcionamento

No diagrama da figura 3.5, observa-se as bases de dados da ferramenta Ikasu, bem como sua arquitetura dividida em interface, controladores e modelo. A base operacional representa o banco de dados com as tabelas do Ryudragon que são atualmente usadas no jogo. A base de análise representa o armazém de dados construído para a análise do Ryudragon. A interface deve mandar os comandos de usuários para os controladores e receber e mostrar os resultados dos experimentos e ações do usuário. Os *controllers* recebem os comandos da interface e selecionam para quem as ações definidas anteriormente devem ser enviadas, além de terem a função de direcionar resultados de ações para as interfaces corretas. O modelo tem como função captar os dados para a base de análise a partir da base operacional usando uma ferramenta de ETL, usar as classes do Yale para executar os experimentos de mineração de dados e implementar outras funcionalidades da ferramenta.

Para a equipe do Ryudragon, o uso de técnicas de análise e mineração de dados serão de grande valia para determinar pontos que passam desapercebidos sem a análise visual ou relações entre dados numéricos do jogo, tais como:

- máximos e mínimos de valores atingidos por variáveis da base de dados em cada era;
- agrupamento de valores da base de dados por origem, por escolhas na árvore tecnológica ou qualquer outro tipo de classificação possível através dos dados da base de análise;
- descoberta de relações do tipo ”cesta de mercado” (BERRY; LINOFF, 2004) usando uma faixa dos pontos de experiência do jogador, as escolhas na árvore tecnológica ou outros atributos classificadores;
- determinar uma medida de desempenho em XP por turnos jogados para cada jogador;
- usar a medida de desempenho para feudos de cada nível para comparar origens, escolhas da árvore tecnológica ou outros classificadores relevantes;
- classificar o comportamento dos jogadores (POWERS, 2006).

O uso das informações citadas acima ajudará a equipe do Ryudragon a caminhar em direção às soluções para problemas como os de equilíbrio de valores de particularidades para origens, valores de pontos de experiência e valores de modificação para a árvore tecnológica citados na seção 3.1. Isto se dará através da medição periódica dos valores e aplicação dos algoritmos de mineração de dados relevantes para cada caso na base de análises.

A ferramenta Ikasu se propõe a oferecer uma série de experimentos pré-determinados sobre a base de análises do Ryudragon, onde o usuário escolhe um experimento e recebe o seu resultado vindo das classes do Yale. Partindo do funcionamento básico do Yale, que monta um experimento a partir de uma árvore de operadores em que a saída de um operador corresponde à entrada do próximo, temos condições de montar experimentos pré-determinados para a ferramenta Ikasu. Para isso, basta criarmos o experimento com a ferramenta Yale e, após, tendo todos os operadores colocados de forma correta na árvore

de operadores, carregarmos o arquivo de configuração deste experimento para o Ikasu. Cada um dos arquivos de configuração de experimentos é carregado e executado pela Ikasu quando um destes for solicitado pelo usuário.

Com a ferramenta pronta para apresentar resultados de experimentos, a equipe do Ryudragon apontou a necessidade de ter uma forma de, em uma mesma interface, acionar os experimentos, visualizar seus resultados e ter controle remoto sobre estas operações. Para atender a estas demandas, uma interface *Web* fora projetada para o Ikasu usando o servidor de aplicações Apache Tomcat. O servidor Tomcat é considerado um *web container* e suas aplicações consideradas módulos dentro deste sistema. A Ikasu foi concebida para funcionar dentro deste molde, em que a aplicação deve ser compilada de forma a obedecer uma estrutura de diretórios padrão, ser transportada para um diretório específico para então ser carregada pelo servidor e funcionar como um sistema *Web*.

3.2.1 Projeto

O projeto da ferramenta Ikasu foi realizado tendo em vista o suprimento das necessidades da equipe de suporte/manutenção do jogo Ryudragon. Algumas das principais necessidades podem ser descritas no diagrama de caso de uso da figura 3.6. No sistema

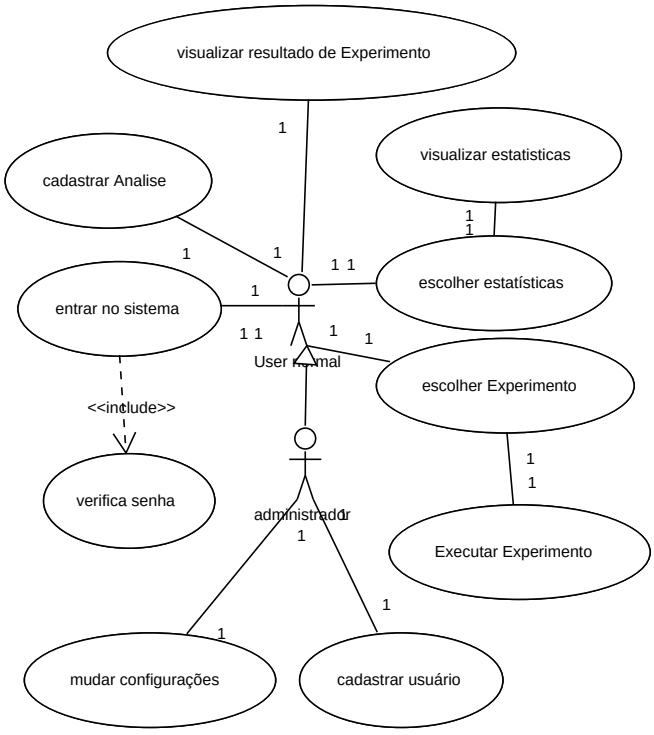

Figura 3.6: Ikasu - Diagrama de casos de uso

Ikasu, existem dois atores possíveis para o uso, os usuários comuns e os administradores.

Como usuário comum, pode-se cadastrar uma nova configuração de experimento no sistema, que consiste principalmente no xml pré-configurado na aplicação Yale, pode-se escolher um dos tipos de experimento, executá-los e acessar os resultados desta execução. O usuário comum também pode escolher e visualizar certas estatísticas sobre a base de análises. Estas estatísticas são consultas pré-definidas pelo Ikasu que retornam ao usuário informações diversas sobre a situação final da era escolhida. Como usuário administrador, além de se ter os privilégios de um usuário comum, pode-se cadastrar outros usuários e se modificar a tabela de configurações do sistema.

Para se alcançar os requisitos descritos acima, o projeto do sistema foi divido em duas partes: a modelagem de uma base de pesquisa que será populada a partir das tabelas principais do Ryudragon e a modelagem do sistema que trabalhará sobre estas tabelas. Estas duas partes são responsáveis por implementar o modelo apresentado na figura 3.5.

A modelagem da base de dados de pesquisa foi dividida em duas etapas. Na primeira etapa, projetou-se uma tabela principal que recebe as informações extraídas das tabelas do Ryudragon (ver tabela 3.1). Ela conta com informações relevantes sobre a situação dos jogadores no final do jogo, como seu nível e escolhas na árvore de tecnologias.

Tabela 3.1: Ikasu - Modelagem da tabela de análise - Primeira modelagem

campo	função
fk user	chave de identificação de usuário e feudo
assinante	mostra se o usuário é assinante ou não
origem	origem escolhida pelo usuário
nivel	níveis do jogador
xp total	pontos de experiência do jogador
arvore tecnologica	código formado pela escolha da árvore tecnológica
Campos de tecnologia	vinte e seis campos binários que dizem se o jogador tem ou não determinada tecnologia
numero era	identificação da era

A segunda etapa da modelagem da base de pesquisa baseou-se no conceito de *Data Marts* e modelo estrela (MOLINA; ULLMAN; WINDOM, 2001). No modelo concebido, existe uma tabela central (tabela de fatos) que guarda as informações relevantes à análise do Ryudragon que podem mudar a cada prospecção feita em sua base dentro de uma mesma era, além de chaves estrangeiras das tabelas de dimensões. Estas tabelas de dimensões possuem dados que não mudam com tanta freqüência ou que são comuns a muitos registros da tabela de fatos. As dimensões criadas são: cadastro-usuário, respon-

sável pelas informações cadastrais do jogador, cadastro-era, responsável por manter as informações sobre as contas que são mudadas apenas no início das eras, cadastro-geo, uma tabela responsável pelas informações geo-cadastrais dos usuários, clãs, responsável pela situação do clã do usuário e geral, responsável por informações comuns a todos os feudos. Este modelo pode ser visto na figura 3.7.

A outra etapa de modelagem da ferramenta Ikasu remete ao sistema que trabalhará sobre as tabelas de análise com o auxílio de outras tabelas. Este sistema foi concebido com base no modelo MVC (*Model, View, Controller*) (WALLS; BREIDENBACH, 2005), que visa separar o sistema em três partes: o modelo, composto pelas classes relativas à finalidade do *software*; a visão, que mostra as informações geradas para o usuário e os controladores, responsáveis por captar a entrada do usuário, usar as classes do modelo e dar informações à visão. Uma representação mais detalhada desta parte do sistema pode ser vista no diagrama da figura 3.8.

Este diagrama de classes apresenta as principais classes do sistema Ikasu. No modelo MVC, a parte do modelo é representada pelos pacotes dao, model e service. O pacote service possui classes responsáveis por cálculos e funções auxiliares requisitadas diretamente pela camada de controladores e são suas classes que fazem a ligação do Ikasu com o Yale. O pacote dao, de *Data Acess Object*, possui classes responsáveis por consultar as bases de dados e o pacote model possui classes que representam as tabelas da base auxiliar e da base de análises. O pacote model e dao são utilizados/utilizam o *framework* Hibernate (BAUER; KING, 2005) para lidar com a base MySQL abstraindo consultas diretas a base de dados. A parte dos controladores é representada pelo pacote web, que recebe as requisições do usuário e as direcionam para a camada model e na resposta, determina quais páginas são mostradas.

3.2.2 Implementação

Nesta seção aborda-se algumas questões gerais de implementação que marcaram o desenvolvimento do projeto. Também discute-se alguns pontos críticos de implementação, particularmente no que se refere à montagem das tabelas de análise a partir da base do Ryudragon. Ao final, sugere-se uma leitura sobre a adaptação da ferramenta Yale para uma aplicação Web.

Como questões gerais de implementação, tem-se a escolha da linguagem de progra-

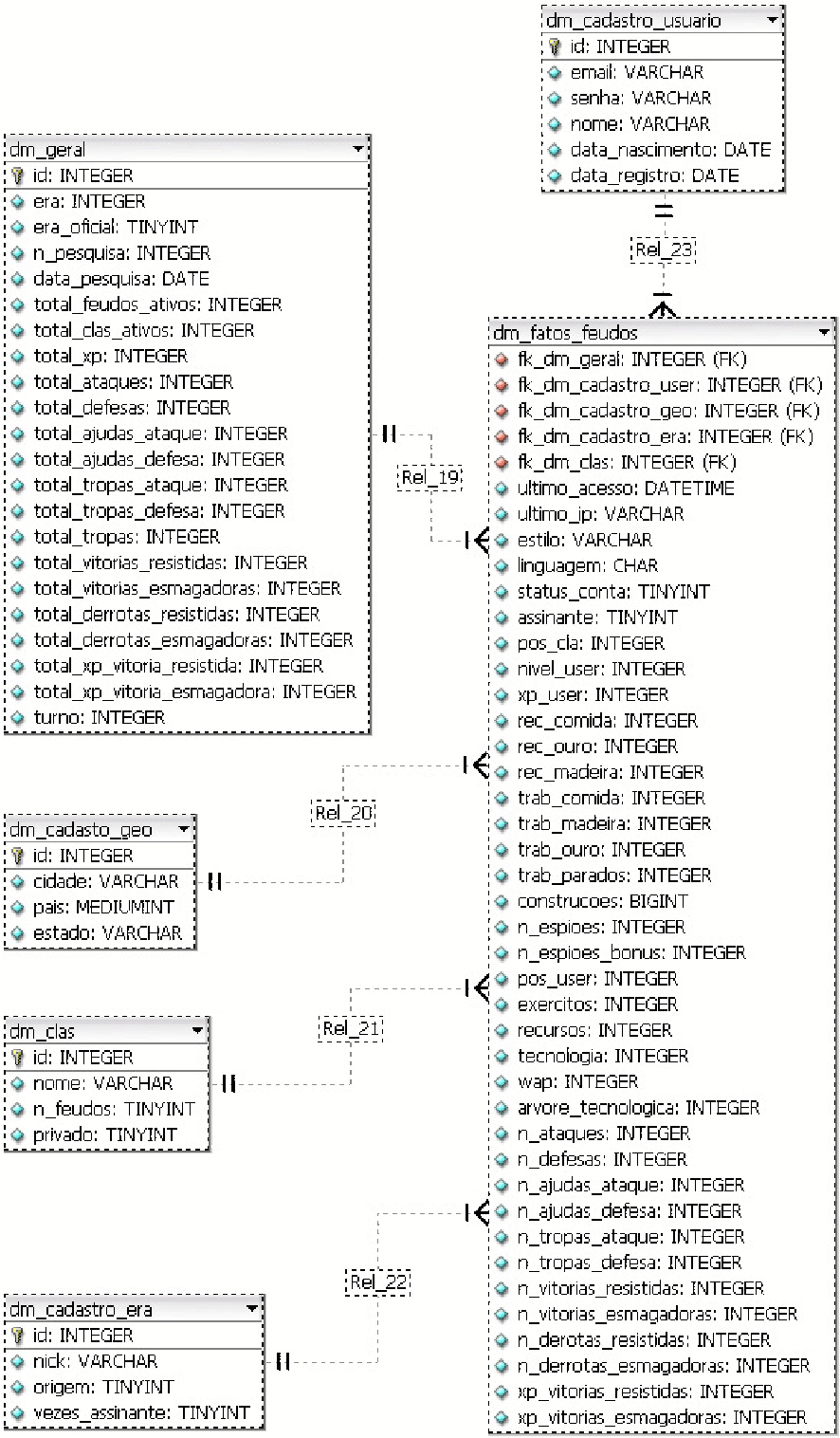

Figura 3.7: Ikasu - Modelagem da tabela de análise - Segunda modelagem

mação usada para o sistema e a escolha da ferramenta de banco de dados. O Ikasu foi programado em Java 1.5 devido ao *framework* Yale ser programado na mesma linguagem, o que dá uma maior facilidade para adaptações e para o uso do mesmo. A ferramenta

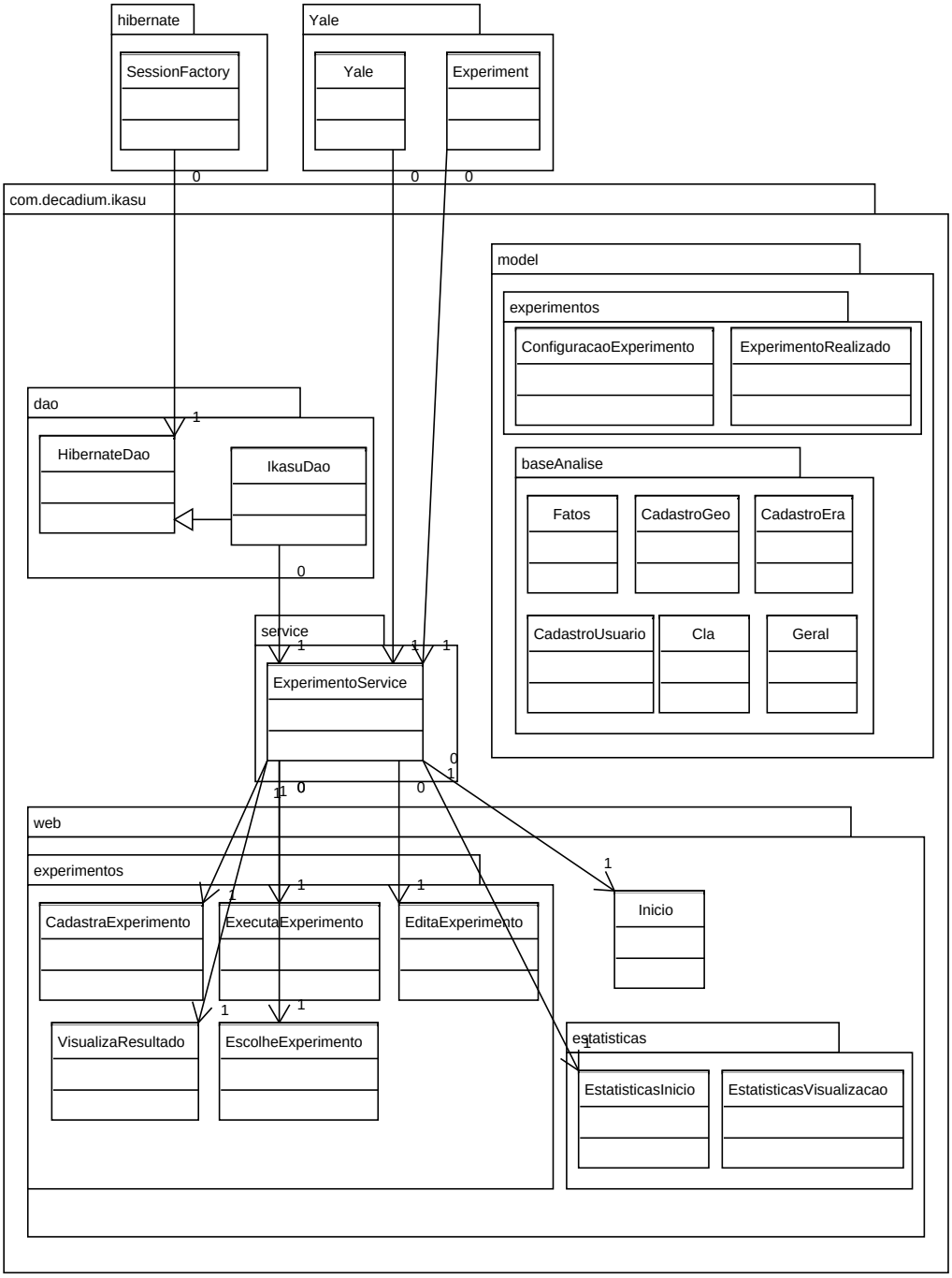

Figura 3.8: Ikasu - Diagrama de classes

de base de dados escolhida foi a MySQL (MYSQL, 1995) devido à base do Ryudragon ser do mesmo tipo, evitando assim possíveis problemas com compatibilidade de tipos de dados e sintaxe de consultas.

As tabelas de análise usadas até o presente momento foram construídas a partir de cópias de segurança da base *on-line* e sem o uso da ferramenta de ETL planejada para o sistema. Esta situação acontece pelos seguintes motivos:

- a política da empresa que hospeda o jogo Ryudragon impede a conexão direta de *software* de terceiros com seus servidores internos. Enquanto esta política não for modificada, a ferramenta de ETL não pode fazer a prospecção dos dados na base ativa do Ryudragon;

- o desenvolvimento de uma ferramenta ETL particular à base de análise é onerosa em termos de tempo de desenvolvimento e não representa o foco principal deste trabalho, sendo interpretada como uma tarefa para o aperfeiçoamento do sistema;

- a Decadium faz regularmente cópias de segurança da base do Ryudragon para uma máquina local;

- a partir da cópia de segurança tem-se uma visão atualizada das bases do jogo e condizente com as necessidades dos experimentos de mineração de dados planejados.

As principais dificuldades encontradas nesta abordagem são os inconvenientes para popular as bases. Para atender aos experimentos planejados, deve-se fazer duas cópias de segurança da base por semana, que compactada atualmente corresponde a cerca de trezentos megabytes de informação que devem ser recuperados via rede. Outro inconveniente está no momento de montar e descompactar a cópia, processo que pode vir a prejudicar o serviço da máquina a outras aplicações, devido ao constante requerimento de entrada e saída em disco rígido que esta ação demanda.

Particularmente para a montagem da base de análise atual, encontrou-se uma dificuldade no momento de se realizar a seleção dos campos vindos da base *on-line*. Existem campos na tabela de análise que não podem ser preenchidos por uma seleção simples vinda da base operacional e devem ser calculados na própria consulta ao banco de dados. Isso resulta no uso do banco com uma tarefa que *a priori* deveria ser resolvida na fase transformação em uma ferramenta ETL.

Outra dificuldade refere-se ao número e tamanho das tabelas onde os dados devem ser captados. Para certas consultas o número de relações inter-tabelas pode comprometer o uso da memória da máquina servidor por parte de outros programas devido ao alto custo de algumas operações de junção. Um exemplo disso ocorreu na prospecção dos campos que se referem ao número de turnos em que o jogador fez pelo menos uma jogada, quantas jogadas de ataque e quantas jogadas de defesa. A seleção feita realizava uma junção de três sub-tabelas da tabela de *logs* do Ryudragon, que possuía trinta milhões de registros.

O resultado foi o processamento cancelado de sete horas da consulta em uma máquina com dois gigabytes de memória física. A solução encontrada para este problema foi a criação de sub-tabelas para que estes campos fossem preenchidos um a um e depois fossem consultados para o preenchimento da tabela de análise principal.

No que se refere à adaptação da ferramenta Yale em um ambiente *on-line*, é importante mencionar que a ferramenta Yale a princípio foi projetada para servir como aplicação *desktop*, sendo extensível e podendo ser usada como um *framework* por outras aplicações. No entanto, seu uso em sistemas para Web ainda não foi explorado pelos construtores do *software*, necessitando assim de adaptações para o uso neste trabalho. Alguns dos pontos relevantes a este assunto são discutidos no primeiro apêndice deste texto.

4 AVALIAÇÃO

4.1 Experimentos usando a metodologia CRISP-DM

Nesta seção veremos um dos experimentos de mineração de dados realizados para a base de dados do Ryudragon da sua sétima era (finalizada em agosto de 2006). Serão explicados os passos seguidos para a realização do experimento, baseados no modelo CRISP-DM e usando o Yale.

Primeiramente, a equipe de desenvolvimento chegou à conclusão de que não se tinha atualmente uma forma concreta de medir o impacto das *features* programadas para o jogo Ryudragon em seu equilíbrio numérico e nem mesmo uma forma segura de analisar ou identificar este estado de equilíbrio. Ou seja, não se tinha uma forma de medir o comportamento dos usuário e suas reações face a mudanças ou inserção de novas características no jogo. Também notou-se que não existia uma forma de análise global e unificada dos dados do jogo para identificar situações que poderiam ser descobertas por EDA (análise exploratória dos dados), nem uma forma de se afirmar que o jogo está próximo do equilíbrio. A partir destes problemas foram definidas metas para serem resolvidas com o uso da mineração de dados:

- criar um processo que possibilite a análise visual e numérica dos dados do Ryudragon;
- estudar o comportamento dos usuários segundo as escolhas na árvore tecnológica;
- definir um consenso sobre o ponto de equilíbrio numérico dentro do jogo e criar um mecanismo que indique este ponto e a partir dele poder analisar o impacto das mudanças no sistema de jogo.

A título de exemplo, será explicado como se fez para atingir o segundo objetivo, o de estudar a **árvore tecnológica**. A estratégia primeiramente pensada para este objetivo foi

identificar e dividir a população de jogadores em estilos de jogo segundo suas escolhas na árvore de tecnologias e, logo após, estudar este comportamento e relacioná-lo com as escolhas de origens e níveis atingidos. Estes estilos de jogo foram definidos como:

- DefensorForte: representado pelas escolhas na árvore que representam melhores jogadores com características de guardar recursos e defender-se de ataques;
- DefensorMedio: escolhas com perfil mais agressivo que o primeiro;
- Engenheiro: com foco nas armas de cerco;
- Guerreiro: escolhas com perfil para atacar de forma maximizada outros feudos;
- NA: um perfil sem classificação.

Outros estilos foram inseridos após uma remodelagem do conjunto de treinamento, o MEPuro ou Multi-estilo Puro, que corresponde a jogadores que optaram por jogar com no mínimo dois dos estilos citados anteriormente e o MEParcial ou Multi-Estilo Parcial, correspondente a jogadores com um estilo definido e que usam parte dos outros. Para realizar tarefas de classificação, Larose (LAROSE, 2005) sugere que se usem algoritmos como árvores de decisão (QUINLAN, 1993), K-vizinhos mais próximos (AHA; KIBLE, 1991) ou redes neurais (REED; II, 1999).

Seguindo os passos da metodologia CRISP-DM, a próxima tarefa um estudo dos dados na base do Ryudragon e seu diagrama entidade-relacionamento. Após, procurou-se por campos considerados relevantes para dividir a população em estilos de jogo, e foram feitas verificações de consistência destes dados.

Como terceiro passo, transferiu-se os dados da base operacional para a base de testes descrita nos capítulos anteriores (figura 3.7) usando seleções diretas dos campos e transformações nos dados.

Na quarta fase, pesquisou-se formas de se realizar um experimento com árvores de decisão e K-vizinhos mais próximos usando o Yale. Encontrando-se o operador apropriado no Yale, criou-se um experimento dentro do *software* contendo a árvore com os operadores relevantes. Criou-se então um conjunto de treinamento para o algoritmo. Este conjunto possui os rótulos com os nomes dos estilos de jogo e os campos com os valores característicos de cada um. Para cada rótulo, foram escolhidos um conjunto típico de várias entradas que eram consideradas pertencentes àquela classe para serem colocadas

no conjunto de treinamento. O conjunto ficou formado por entradas na base de jogadores com nível maior que seis para se caracterizar mais os representantes dos estilos. Foram gerados modelos usando os operadores de árvores de decisão J48 (QUINLAN, 1993), id3 (QUINLAN, 1986) e REPtree (WITTEN; FRANK, 2005) e o operador de K-vizinhos mais próximos IBk (AHA; KIBLE, 1991). Após isto, os algoritmos foram treinados com o conjunto de treinamento e aplicou-se o modelo criado nos dados da base de análises.

A quinta fase serviu para testar a concordância dos modelos criados no Yale com o modelo do Ryudragon. Voltou-se muitas vezes para a quarta fase até se consolidar um operador que resultasse em um modelo próximo ao objeto de estudo, inclusive com remodelagens no conjunto de treinamento resultantes de uma volta até a primeira fase da metodologia.

Após, na última fase da experiência e com os modelos aprovados, foram criados gráficos e tabelas para análise dos resultados obtidos.

4.2 Casos de uso da ferramenta Ikasu

A ferramenta Ikasu teve sua avaliação feita mediante ao seu atendimento aos casos de uso requisitados no planejamento. Para iniciar o funcionamento do Ikasu, o sistema deve estar devidamente compilado e inserido em um servidor de aplicação Tomcat 5.1 ativo. Também deve-se certificar do funcionamento da base de dados MySQL onde se encontram as tabelas de análise e as tabelas auxiliares. Nesta avaliação, primeiramente veremos como foram tratados a entrada de usuários no sistema e o cadastro de experimentos pré-configurados. Após, veremos a metodologia de testes e avaliação para se escolher e visualizar estatísticas do Ryudragon e por último, como foram avaliadas a escolha e execução dos experimentos pré-configurados.

Ao acessar a URL do Ikasu através do navegador de internet, o usuário se encontra na tela de *login*, onde deve digitar seu nome de usuário e senha para ter acesso às outras telas do sistema. Ao efetuar a entrada no sistema, o usuário tem acesso ao menu principal, onde pode escolher as ações a serem efetuadas no sistema. Através deste menu, o usuário pode escolher a opção de cadastrar um novo experimento no sistema. Através de um formulário, o usuário preenche os campos necessários e o envia. Ao enviar o formulário, o sistema passa a requisição para a classe controladora responsável, que valida a entrada e grava os dados na base usando a classe de serviço de experimentos. Esta classe de serviço

usa métodos do *Data Access Object* (DAO), ligado ao *framework* Hibernate para propagar as mudanças na base.

Outra opção do usuário é a de acessar as Estatísticas de uma determinada era do Ryudragon. O usuário preenche os dados no formulário da página *web* relevantes, mostrado na figura 4.1, como número da era e quais estatísticas quer gerar, e após, envia a requisição.

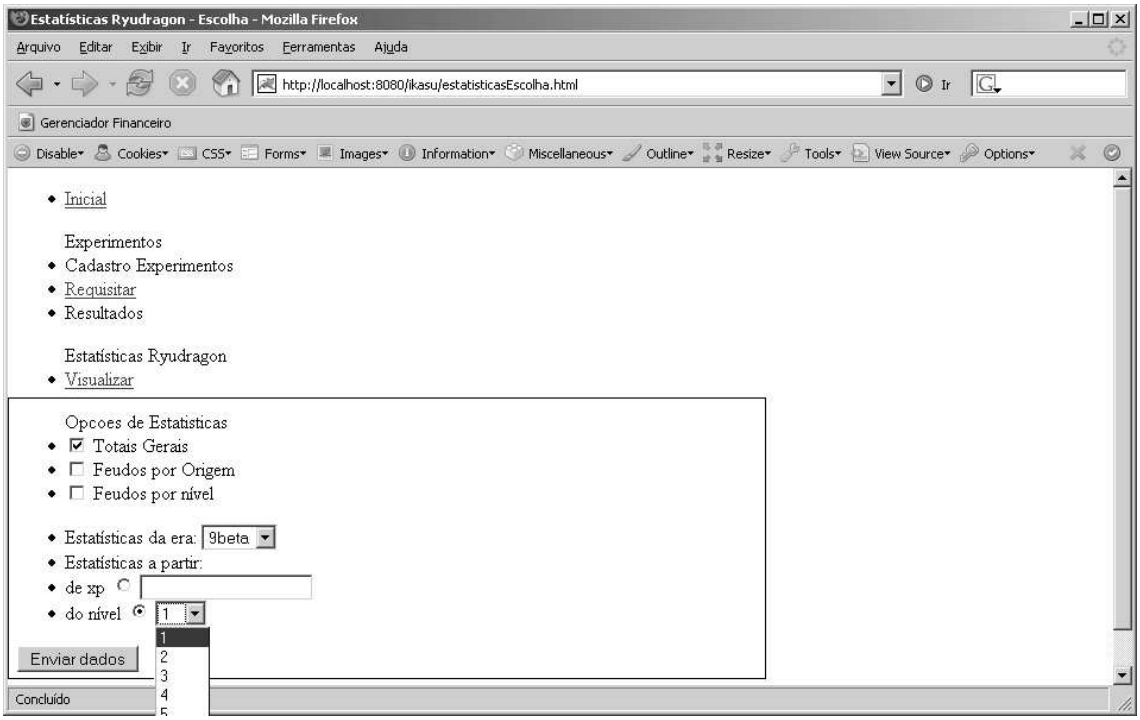

Figura 4.1: Ikasu - seleção de estatísticas

O formulário, assim como no caso anterior, chega ao controlador responsável, que o valida e usa seus parâmetros na classe de serviço e DAO para efetuar a consulta na base de análises. Esta consulta retorna uma página com a visualização das estatísticas requisitadas mostrada na figura 4.2. Os dados do resultado destas consultas é semelhante ao das tabelas de EDA vistas na próxima seção.

Outra opção que funciona de forma semelhante a anterior é a de executar um experimento pré-configurado através da interface mostrada na figura 4.3.

Ao se escolher esta opção no menu, o controlador responsável busca os experimentos cadastrados na base de dados e os mostra como opção de escolha ao usuário. O usuário escolhe um deles e manda executá-lo. O controlador responsável capta a requisição com os dados do experimento e usa o serviço de experimentos para executar o comando. A partir deste ponto, no serviço de experimentos, o Yale deveria ser inicializado, ter o ar-

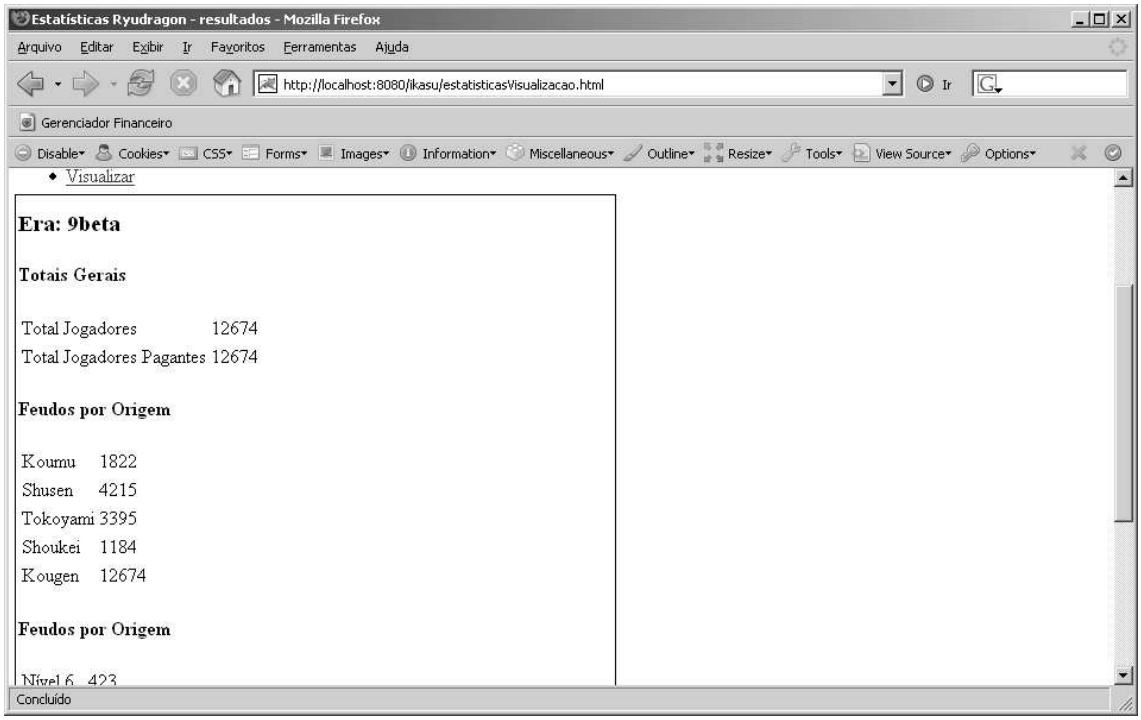

Figura 4.2: Ikasu - resultado de estatísticas

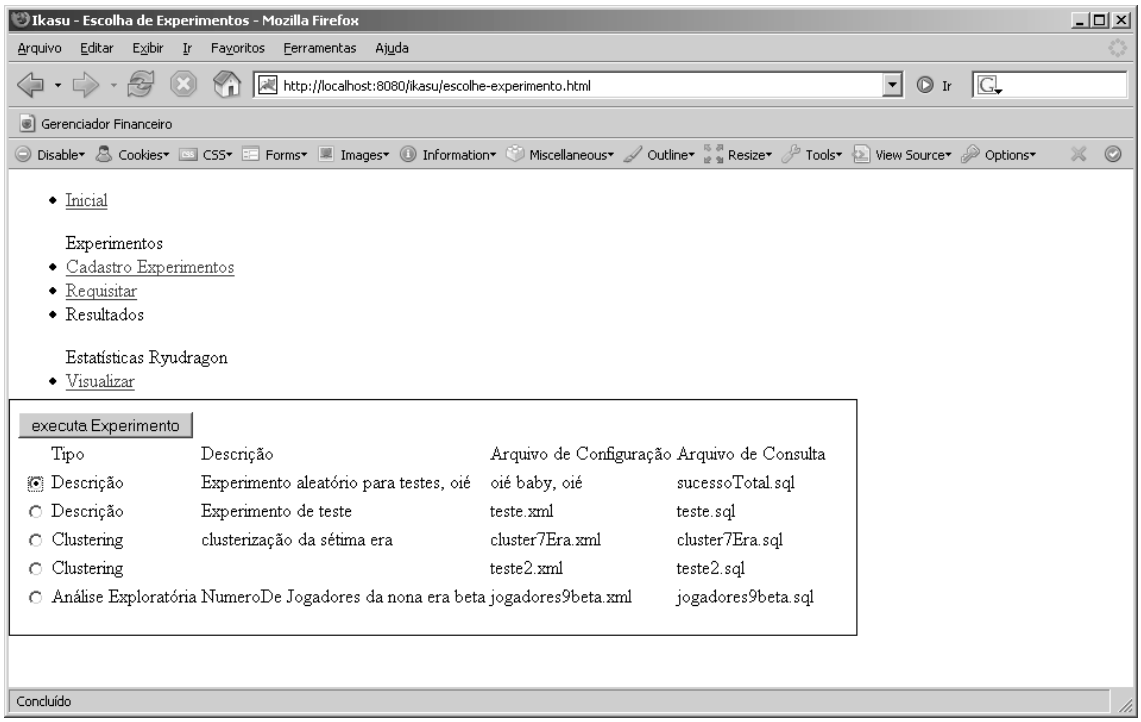

Figura 4.3: Ikasu - seleção de experimentos

quivo de experimento pré-configurado carregado e então ter o experimento executado. A saída deveria ser gravada em um diretório pré-determinado para que o usuário pudesse visualizar quando entrasse no sistema. Mas o que acontece quando mandamos executar o experimento no Ikasu é uma falha, onde o Yale não passa da inicialização de atributos e

fica carregando por tempo indeterminado as classes de operadores. O experimento com o uso do Yale em uma aplicação não- *web* funcionou corretamente, mas não funcionou até o momento na Ikasu.

4.3 Visualização

Com a realização de experimentos de uma tarefa de descrição na base de dados do Ryudragon, a equipe conseguiu informações relevantes sobre a situação do jogo no final da sétima era, setembro a outubro de 2006 e da nona era de beta-teste em dezembro de 2006. A análise foi feita sobre a situação final do jogo onde os jogadores alcançaram seus pontos máximos em experiência, níveis e suas escolhas nas árvores tecnológicas. O total de jogadores considerados para a análise consistiu nas contas que se encontravam a partir do primeiro nível e que atingiram pelo menos novecentos pontos de experiência. O uso destas contas, chamadas de contas válidas, serviu para retirar da análise feudos que foram criados mas não foram usados dentro do jogo. A seguir serão apresentados os resultados obtidos pela análise exploratória dos dados do Ryudragon e os dados obtidos na tentativa de classificar os jogadores do Ryudragon em estilos.

A análise exploratória usou tanto conteúdo numérico prospectado com consultas diretas ao banco de dados, quanto a gráficos de apoio gerados pelo Yale. A partir da exploração dos valores, notamos que, primeiramente, o número de contas válidas tanto na sétima, quanto na era beta apresenta-se em torno da metade do total de contas (46% de contas válidas na sétima era e 52% na era beta). Em segundo lugar, nota-se que não há o equilíbrio desejado pela equipe do Ryudragon na escolha das origens por parte dos jogadores, como mostrado na tabela 4.1.

Tabela 4.1: Ryudragon - Disposição de feudos válidos por origens

		Sétima Era			Beta Nona Era
	Total	Percentagem		Total	Percentagem
Feudos Válidos	12987	-		6661	-
Total de Kougen	2743	21		1025	15
Total de Koumu	1748	13		950	14
Total de Shoukei	1398	11		735	11
Total de Shusen	3459	27		2363	35
Total de Tokoyami	3639	28		1588	24

A tabela mostra uma defasagem na escolha da raça dos Shoukei e dos Koumu em relação às outras, sendo que a menos escolhida corresponde a menos da metade dos pontos

percentuais da mais escolhida nas duas eras. O fato do quadro se manter aproximado em relação às origens menos escolhidas nas duas eras apresenta a possibilidade de um problema constante no decorrer das eras.

Como segundo resultado da análise exploratória realizada, tem-se que o número de jogadores pagantes tem preferência pela origem dos Kougen e quase não escolhem os Koumu e os Shoukei. Como visto na figura 4.4, os pontos em vermelho, que representam os jogadores pagantes, aglomeram-se em maior número na origem um, os Kougen.

Os jogadores pagantes não recebem vantagens nos seus valores e cálculos dentro de jogo, mas recebem vantagens adicionais como poder escolher em que time jogarão. Esta parcela da população de jogadores representa, em sua maioria, jogadores experientes e dedicados ao jogo, sendo de suma importância para a equipe de desenvolvimento o monitoramento de seu comportamento dentro do Ryudragon. No caso deste descompensamento entre as escolhas das origens, não se tem uma explicação clara até o momento para tal comportamento.

Um terceiro ponto levantado pela análise exploratória remetente ao gráfico acima diz respeito à possível dificuldade que os jogadores não pagantes, representados em azul no gráfico 4.4, têm em relação aos pagantes para chegar aos níveis mais elevados dentro do jogo. Nota-se que, em todas as origens, a presença de pontos azuis é quase inexistente nos níveis mais elevados que são os de oito a dez.

Os principais demonstrativos do resultado do experimento de mineração de dados realizado para classificar os jogadores segundo suas escolhas na árvore tecnológica, será exposto a seguir. Primeiramente, foram executados os algoritmos de árvores de decisão e K-vizinhos mais próximos sobre o próprio conjunto usado no treinamento com algumas entradas repetidas. Os algoritmos que melhor se apresentaram foram os de REPTree e o J48, apesar de não alcançarem um resultado ótimo, foram os que mais se aproximaram dos números do conjunto de avaliação, como mostrado na tabela 4.2.

Tabela 4.2: Estilos de jogo - Resultado dos algoritmos em um conjunto de validação

	Engenheiro	Guerreiro	Defensor Médio	Defensor Forte
Conjunto de Avaliação	11	105	122	335
REPTree	10	105	123	335
ID3	105	11	122	335
J48	11	105	122	335
K-vizinhos	174	47	36	316

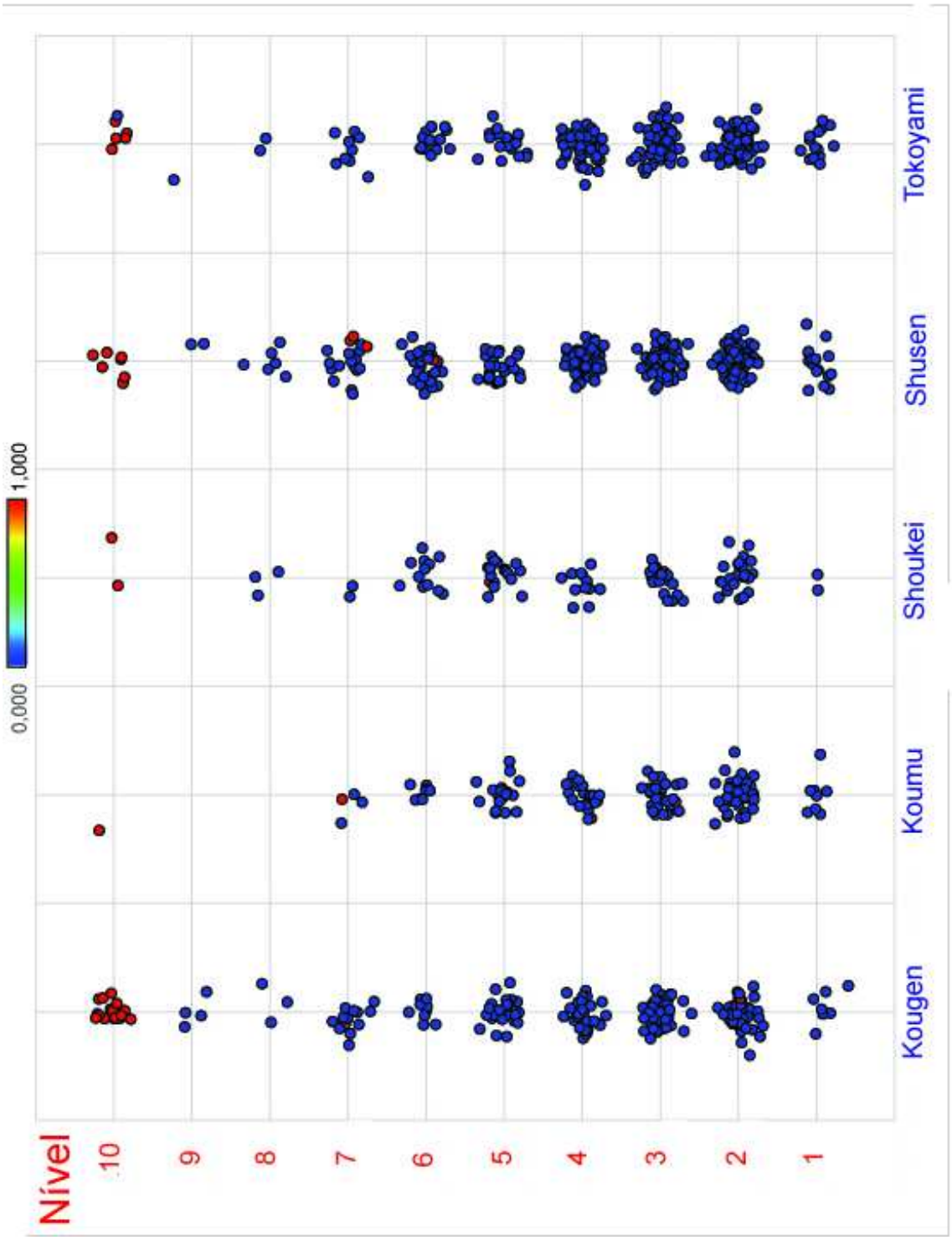

Figura 4.4: Ryudragon - Gráfico de nível (eixo y) por origem (eixo x), Os pontos representam os jogadores, sendo os azuis os que jogam gratuitamente

Após esta avaliação, foi realizada a inserção dos estilos de multi-estilo e o de estilo indefinido (NA). Então, com os mesmos modelos usados anteriormente, foram realizadas experiências sobre dados de jogadores que estivessem acima do nível seis. Os algoritmos que melhor se saíram foram o REPTree e o K-vizinhos mais próximos, por terem seus

Tabela 4.3: Estilos de jogo - Resultado dos algoritmos nos jogadores de nível 7+

	Eng	Guer	Def Médio	Def Forte	ME Puro	ME Parc	NA
Conjunto de Treinamento	7	17	27	29	11	60	20
REPTree	15	470	127	354	33	320	93
ID3	91	84	13	36	278	878	32
J48	2	364	0	1	46	917	82
K-vizinhos	42	594	149	120	24	446	37

resultados mais aproximados com os valores do conjunto de treinamento, como mostrado na tabela 4.3.

Os modelos gerados pela ID3 e pelo algoritmo J48 não serão usados, pois no ID3 foram encontrados menos defensores médios que no conjunto de treinamento e no J48 o número de engenheiros, defensores médios e defensores fortes ficaram abaixo dos colocados no conjunto de treinamento. Nota-se também que no algoritmo de REPTree houve uma sobrecarga nos valores dos guerreiros, pois através da consulta direta, obteve-se cento e cinco usuários que se enquadravam nesta categoria contra os quatrocentos e setenta indicados pelo algoritmo. As classificações errôneas dos guerreiros vieram da classe NA, pois por seleção direta foram encontrados quinhentos e doze jogadores que se enquadravam neste estilo contra noventa e três indicado pelo algoritmo. O mesmo nota-se no resultado apresentado pelo algoritmo dos K-vizinhos mais próximos.

Como base no resultado escolhido, o do algoritmo REPTree, e sabendo que os guerreiros estão sobrecarregados e os NA sub-carregados, foi gerado o gráfico apresentado na figura 4.5, que explica a distribuição dos estilos em relação as origens escolhidas.

Nota-se que, visualmente, as escolhas de estilo em uma origem estão equilibradas em densidade gráfica com a mesma escolha nas outras origens. Este equilíbrio só não ocorre com o estilo engenheiro por ter poucos exemplares e com o estilo NA por não ter sua população corretamente classificada.

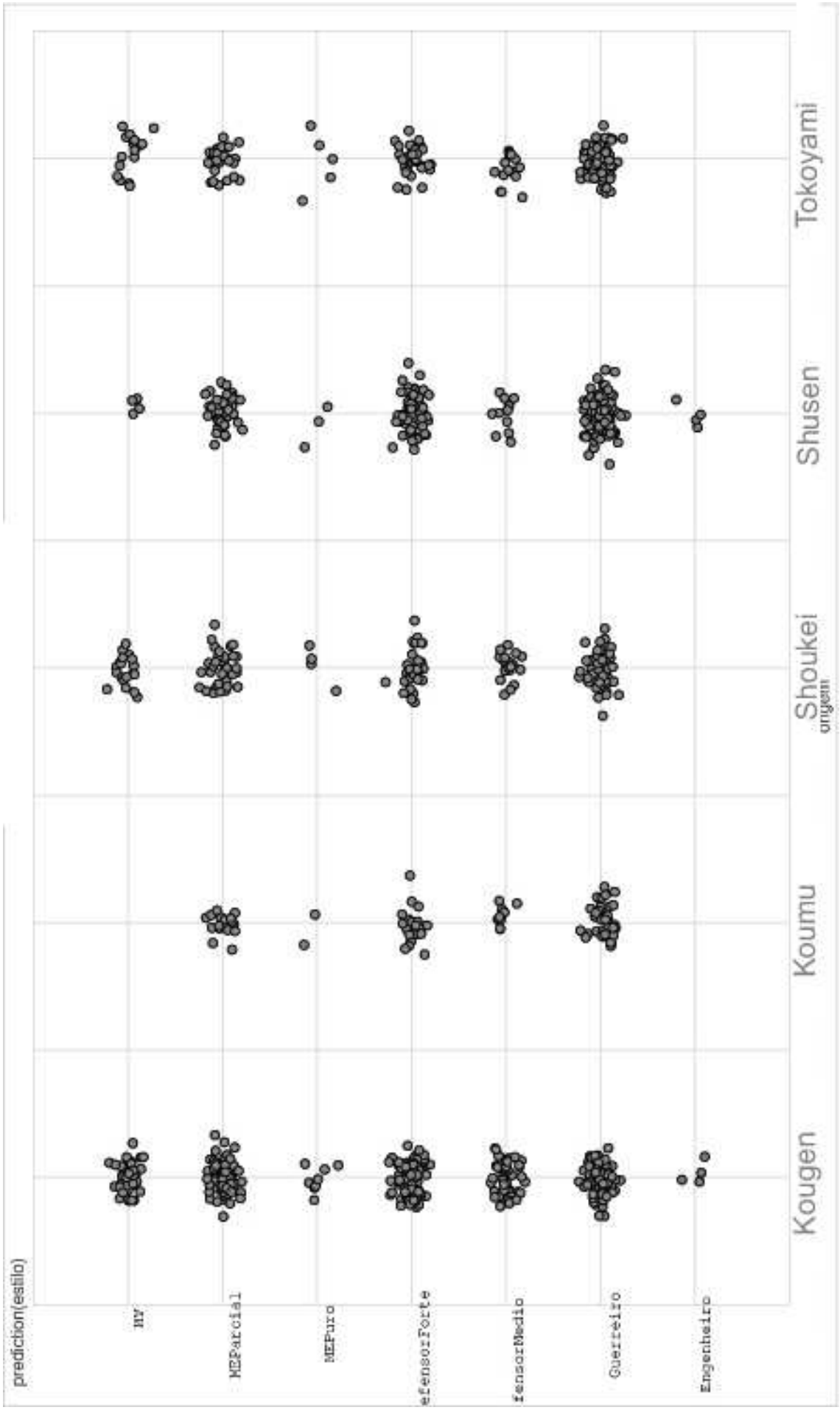

Figura 4.5: Estilos De Jogo - Estilos escolhidos em cada Origem

5 CONCLUSÃO

5.1 Considerações Finais

O uso das técnicas de mineração de dados na base do Ryudragon mostraram-se de valia para a equipe de desenvolvimento do jogo por proporcionar a elucidação de pontos obscuros dentro do sistema e possibilitar idéias para a análise de possíveis pontos de falha no equilíbrio numérico do jogo. Isto traduz-se de forma concreta até o presente momento nos resultados apresentados na análise exploratória e nos testes realizados com o Yale. Estas experiências possibilitaram a equipe iniciar uma análise construtiva do comportamento dos dados do Ryudragon para formar uma idéia clara do posicionamento de certos elementos dentro do universo do jogo, levantando assim, algumas questões a serem resolvidas a priori. Destas, um primeiro ponto a ser abordado é o fato de que não há o equilíbrio desejado pela equipe de desenvolvimento do jogo na escolha das origens pelos usuários. Como mostrado no capitulo anterior, a população de feudos válidos está dando preferência maior a escolhas de uma ou duas origens.

Um segundo ponto a ser investigado, é a dificuldade que os jogadores não pagantes têm para chegar as níveis mais altos dentro do jogo e a baixa taxa de escolha apresentada para as tecnologias do estilo engenheiro.

Outro ponto que pode ser abordado diz respeito a escolha e aperfeiçoamento de um modelo para classificação dos jogadores segundo suas escolhas na árvore de tecnologias. O erro observado na precisão dos algoritmos de mineração de dados neste caso pode ter sido ocasionado por uma má escolha no conjunto de treinamento (LAROSE, 2005), que pode não ter representado os estilos da forma como deveria para que o resultado desejado fosse alcançado. Outra explicação pode se dar pela grande quantidade de campos a serem memorizados (26 tecnologias) e a baixa quantia de exemplos para cada classe, totalizando pouco mais de trinta combinações de escolhas para cada estilo. Este baixo número de

exemplos pode ter feito com que os modelos criados pelos algoritmos resultassem de forma errônea quando aplicados no conjunto de dados real.

Do presente trabalho também podemos concluir que o uso da ferramenta Yale 3.4 como biblioteca de funções para uma aplicação *web* em um servidor de aplicações Apache Tomcat 5.1 apresentou-se como um impedimento para a conclusão esperada para o mesmo. Isto deve-se ao fato de que o Yale 3.4 não foi projetado para funcionar com uma aplicação web nos moldes dos *web containers* do Tomcat. Mais detalhes sobre o funcionamento do Yale e sugestões para sua integração a uma aplicação *web* podem ser vistos no primeiro apêndice deste texto.

Ao concluir este projeto, têm-se que, com a proposta do uso da mineração de dados no jogo Ryudragon e o estudo para sua implantação, a equipe de desenvolvimento e manutenção do jogo Ryudragon adquiriu benefícios em termos de gestão de jogos. A pesquisa para a obtenção de uma forma de se chegar ao equilíbrio numérico no jogo desenvolveu na equipe do Ryudragon uma noção de que as escolhas do jogador são baseadas em situações que lhes tragam mais vantagens, e isso se reflete no comportamento dos dados do jogo (KENNERLY, 2003). Com o estudo para a implementação de um *data mart* para o Ryudragon, a equipe necessitou analisar, corrigir e otimizar tabelas do Ryudragon e preparar-se para abrigar uma infra-estrutura de um sistema de apoio a decisão. A análise exploratória dos dados do Ryudragon elucidou pontos que não eram claros e apresentou informações que não tinham sido consideradas de antemão como formas de medir a situação atual do jogo em termos de equilíbrio numérico do sistema. Estas informações servem de base inicial para hipóteses a serem estudadas e trabalhadas através de um processo cíclico de mineração de dados.

5.2 Trabalhos Futuros

A partir deste ponto no projeto, existem questões que podem ser levantadas para serem resolvidas em um momento posterior. A primeira delas diz respeito a criação e adaptação de um armazém de dados para a Decadium Studios. Este armazém além de conter as informações gerenciais da empresa e de outros jogos produzidos pela mesma, conterá as informações sobre o jogo Ryudragon que são utilizadas no presente trabalho que serão transportadas e adaptadas para a nova base.

A criação deste armazém de dados levará a uma segunda questão, que diz respeito

à construção de um *data mart* novo para o Ryudragon, baseado na nova estrutura e o planejamento de uma respectiva ferramenta de ETL. O processo todo engloba rever os dados necessitados para as análises, construir as novas tabelas, construir uma ferramenta de ELT e povoar o *data mart*.

Uma terceira questão que pode ser discutida é o aprofundamento nos algoritmos de mineração de dados. Esta questão levanta a necessidade de mais estudo sobre as possibilidades de análises feitas sobre a base de dados do Ryudragon e para cada nova técnica, a criação de um experimento no Yale. Uma destas possibilidades é repetir e avaliar o experimento de classificação dos jogadores por suas escolhas na árvore tecnológica usando as Redes de Bayes (LAROSE, 2006), outros algoritmos de árvores de decisão ou outros algoritmos relevantes. Em uma sugestão mais pontual, um próximo estudo sugerido é o de medir os valores de experiência do jogo semanalmente e formar assim uma medida de desempenho para os jogadores, como proposto por Kennerly (KENNERLY, 2003). Esta medida de desempenho poderá ser cruzada com as informações de nível de jogador, origem escolhida e estilo de jogo usado proveniente das informações vindas da análise da árvore de tecnologias. Assim, a equipe do Ryudragon poderá ter uma idéia sobre o desempenho dos jogadores que escolhem um certo tipo de estilo de jogo e assim fazer suposições e indicar medidas possíveis para igualar os desempenhos de cada estilo.

Um outro ponto a ser ressaltado como um trabalho futuro consiste na implementação dos casos de uso faltantes e na correção dos problemas ocorridos na tentativa de se adaptar o Yale à interface web. Os casos de uso que podem ser citados são os da alçada administrador, cadastro de usuários e modificação na tabela de configurações do sistema e os de usuário comum, executar Experimento e visualizar seu resultado. Um ponto inicial para as correções na adaptação do Yale pode vir da depuração intra-framework e na contactação de seus desenvolvedores. Em termos de aperfeiçoamento do sistema *web* podemos citar o melhoramento da interface de usuário de modo a torná-la mais intuitiva e a extensão da ferramenta Ikasu para análises gerenciais e para outros jogos da empresa.

REFERÊNCIAS

AHA, D.; KIBLE, D. Instance-based learning algorithms. **Machine Learning**, [S.l.], v.6, p.37–66, 1991.

APPELCLINE, S. **Designing Strategy**: victory & balance. Disponível em: <http://www.rpg.net/news+reviews/columns/virtually07mar03.html>. Acesso em: 04 de janeiro de 2007, RPGnet.

BAUER, C.; KING, G. **Hibernate in Action**. [S.l.]: Manning Publications Co., 2005.

BERRY, M.; LINOFF, G. **Data Mining Techniques**: for marketing, sales, and costumer relationship management second edition. 2.ed. [S.l.]: Wiley Publishing Inc, 2004.

BONIFATI, A.; CATTANEO, F.; CERI, S.; FUGGETTA, A.; PARABOSCHI, S. Designing Data Marts for Data Warehouses. **ACM Transactions on Software Engineering and Methodology**, [S.l.], v.10, n.4, p.452 – 483, Outubro 2001.

BOSSER, A. G. Massively multi-player games: matching game design with technical design. In: ACM SIGCHI INTERNATIONAL CONFERENCE ON ADVANCES IN COMPUTER ENTERTAINMENT TECHNOLOGY TABLE OF CONTENTS, 2004., 2004, Singapore. **Proceedings...** ACM Press New York: NY: USA, 2004. v.74, p.263–268.

BREIMAN, L.; FRIEDMAN, J.; OLSHEN, R.; STONE, C. **Classsification and Regression Trees**. [S.l.]: Chapman & Hall / CRC Press, 1984.

BUCHANAN, E.; ESS, C. Social Context in Massively-Multiplayer Online Games (MMOGs): ethical questions in shared space. In: INTERNATIONAL REVIEW OF INFORMATION ETHICS, 2005, USA. **Anais...** [S.l.: s.n.], 2005. v.4, n.4. Disponível em: <http://www.i-r-i-e.net/issue4.htm>. Acesso em: 14 de outubro de 2006.

CARPENTER, A. **Applying Risk Analysis To Play-Balance RPGs**. Disponível em: <http://www.gamasutra.com/features/20030611/carpenter_01.shtml>. Acesso em: 04 de janeiro de 2007, Gamasutra.

CHEN, S.; HAN, J. Data Mining: an overview from a database perspective. **IEEE Transactions on Knowledge and Data Engineering**, [S.l.], v.6, p.866–883, 1996.

CHICK, T. **MMOs**: building whole societies, the history of massively multiplayer online games. Disponível em: <http://archive.gamespy.com/amdmmog/week5/>. Acesso em: 14 de outubro de 2006.

DEMSAR. **Orange**: from experimental machine learning to interactive data mining. [S.l.]: Faculty of Computer and Information Science, University of Ljubljana, 2004.

FALOUTSOS, C.; LIN, K. **FastMap**: a fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. Zurich, Switzerland, 1995. 163p. v.1.

FISCHER, S.; KLINKENBERG, R.; MIERSWA, I. **YALE 3.4**: yet another learning environment - user guide, operator reference, developer tutorial. [S.l.: s.n.], 2006.

GAMESPOT. **Game industry study predicts massive growth ahead**. Disponível em: <http://www.gamespot.com/news/6099584.html>. Acesso em: 13 de outobro de 2006, Portal Gamespot.

HAND, D.; MANNILA, H.; SMYTH, P. **Principles of Data Mining**. [S.l.]: The MIT Press, 2001.

IBM. **Business Intelligence Concepts**. Disponível em: <http://www-03.ibm.com/servers/eserver/iseries/db2/dataware.htm>. Acesso em: outubro de 2006, IBM.

IMMON, W. H. **Building the Data Warehouse**. 3.ed. USA: Wiley, 2002.

JúNIOR, J. F. R. **Desenvolvimento de um Framework para análise visual de informações suportando data mining**. 2003. Dissertação (Mestrado) — Instituto de Ciências Matemáticas e de Computação - ICMC - Universidade de São Paulo - USP - São Carlos-SP.

JUKIC, N. Modeling Strategies and alternatives for data warehousing projects. **Communications of the ACM**, [S.l.], v.49, n.4, p.83 – 89, April 2006. Publisher: ACM PRess.

KENNERLY, D. **Better Design through Data Mining**. Disponível em: <http://www.gamasutra.com/features/20030815/kennerly_01.shtml>. Acesso em: 13 de outubro de 2006, USA: Gamasutra.

KIMBALL, R. **The Data Warehouse Lifecycle Toolkit**. EUA: Wiley, 1998.

KIMBALL, R.; ROSS, M. **The data Waharehouse Toolkit**: the complete guide to dimensional modeling. 2.ed. [S.l.]: Wiley, 2002.

KOHONEN, T. Self-organized formation of topologically correct feature maps. **Biological Cybernetics**, [S.l.], v.43, p.59, 1982.

LAROSE, D. T. **Discovering Knowledge in Data - An Introduction to Data Mining**. 1.ed. [S.l.]: John Wiley & Sons Inc., 2005.

LAROSE, D. T. **Data Mining Methods and Models**. 1.ed. [S.l.]: John Wiley & Sons Inc., 2006.

LTDA, D. S. **Wikidragon**. Disponível em: <http://www.ryudragon.com/wiki/>. Acessado em 19 de dezembro de 2006.

MIERSWA, I.; WURST, M.; KLINKENBERG, R.; SCHOLZ, M.; EULER, T. **YALE**: rapid prototyping for complex data mining tasks. Philadelphia, USA, 2006.

MOLINA, H.; ULLMAN, J.; WINDOM, J. **Implementação de sistemas de banco de dados**. Rio de Janeiro, RJ: Prentice Hall, 2001.

MOSS, L.; ATRE, S. **Business Intelligence Roadmap**. EUA: Addison-Wesley, 2003.

MYSQL. **The world's most popular open source database**. Disponível em: <http://www.mysql.com/>. Acesso em: 14 de janeiro de 2007, MySql Web Page.

NEURAL SMITHING: SUPERVISED LEARNING IN FEEDFORWARD ARTIFICIAL NEURAL NETWORKS, 1999, Cambridge, MA. **Anais...** [S.l.: s.n.], 1999.

PALISADE. **Palisade Corporation**. Disponível em: <http://www.palisade.com/risk/default.asp>. Acesso em: 04 de janeiro de 2007, Palisade Corporation.

PATERLINI, A. A.; FARIA, R. F. T. de; RAZENTE, H. L.; JúNIOR, C. T.; TRAINA, A. J. M. **FastMapDB**: uma ferramenta para visualização em sgbdrs com suporte a filtragem e seleçãoo visual dos dados. Av. João Naves de Ávila, 1331 - Piso C - Uberlândia - MG, 2005.

PETE CHAPMAN (NCR) JULIAN CLINTON (SPSS), R. K. N.; (SPSS), T. K.; (DAIMLERCHRYSLER), T. R.; (SPSS), C. S.; (DAIMLERCHRYSLER), R. W. **CRISP DM 1.0 - CRoss Industry Standard Process for Data Mining 1.0**. 1.0.ed. [S.l.]: CRISP-DM Consortium, 2000.

PONNIAH, P. **Data Warehousing Fundamentals**. EUA: Wiley, 2001.

POWERS, S. **It's All About the Audience**. Disponível em: <http://www.skotos.net/articles/lessons/lessons4.phtml>. Acesso em: dezembro de 2006, Skotos.

QUINLAN, R. Induction of decision trees. **Machine Learning**, [S.l.], v.1, p.81–106, 1986.

QUINLAN, R. **C4.5**: programs for machine learning. San Mateo, CA: Morgan Kaufmann Publishers, 1993.

TATSUMOTO, Y.; THAWONMAS, R. MMOG Player Classification Using Hidden Markov Models. In: ENTERTAINMENT COMPUTING (EINDHOVEN, 1-3 SEPTEMBER 2004) ICEC 2004 : INTERNATIONAL CONFERENCE ON ENTERTAINMENT COMPUTING NO3, 2004, Eindhoven. **Anais...** Springer: Berlin: ALLEMAGNE (2004), 2004. v.3166, p.429–434.

TWIKI. **Data Ming - All data mining software**. Disponível em: <http://www.the-data-mine.com/bin/view/Software/AllDataMiningSoftware>. Acesso em: 05 de janeiro de 2007, Twiki.

WALLS, G.; BREIDENBACH, R. **Spring in Action**. [S.l.]: Manning Publications Co., 2005.

WAN, E. E.; XU, X.; ZHOU, Z. **2006 Online Game Report**: pre-release: january 20, 2006. [S.l.]: P PACIFIC ACIFIC EPOCH, 2006.

WITTEN, I.; FRANK, E. **Data Mining**: practical machine learning tools and techniques. 2.ed. San Francisco - USA: Morgan Kaufmann, 2005.

ANEXO A PARTICULARIDADES DO FRAMEWORK YALE

A.1 Formas de Distribuição

O framework Yale, é disponibilizado em sua forma principal como uma aplicação *desktop* possuindo uma interface gráfica que possibilita o usuário visualizar e modificar os elementos de um dado experimento. A outra forma que o Yale é disponibilizado consiste no na sua biblioteca principal em Java e de outras bibliotecas auxiliares, para o uso em aplicações específicas. A seguir serão explanadas algumas peculiaridades do uso do Yale como uma biblioteca de aplicação.

O uso do Yale como ferramenta se dá através de sua instalação na máquina do usuário e o uso de sua interface gráfica, ou por linha de comando para executar os experimentos. No outro modo de usá-lo, temos-lo como uma biblioteca integrada ao *software*. Para ser usado deste modo, deve-se inserir a biblioteca no projeto respeitando uma hierarquia de diretórios fixa, como visto na tabela A.1.

Tabela A.1: Hierarquia de diretórios da ferramenta Yale

etc/	Arquivos de configuração
lib/	biblioteca Java e arquivos JAR
lib/yale.jar	Arquivo principal da biblioteca Yale
resources/	arquivos de recursos requeridos para o funcionamento da biblioteca

Esta hierarquia deve ser respeitada por qualquer aplicação que tenha a intensão de usar o Yale como biblioteca externa.

A.2 Inserção do Yale em uma aplicação Web

Na atual distribuição da ferramenta Yale, versão 3.4, o seu uso como framework *stand-alone* é dificultado por algumas peculiaridades exigidas para o seu funcionamento neste

modo, em especial para as aplicações web que usam um programa servidor Apache Tomcat 5.1. Este fato se dá devido as exigências que o Tomcat tem para que uma aplicação funcione como um de seus módulos, ou *web containers*. Sendo a ferramenta Ikasu projetada para funcionar como um módulo do Tomcat, algumas adaptações no uso do Framework Yale devem ser cogitadas para seu bom funcionamento integrado à referida ferramenta.

Para se compreender as adaptações futuramente sitadas, é importante observar dois pontos. O primeiro ponto é que a ferramenta Yale não funciona por si só como uma biblioteca, Ou seja, as funções do Yale não funcionam em um projeto se apenas o arquivo de funções principal, o yale.jar, for adicionado à compilação. O Yale necessita de outras bibliotecas que não fazem parte da distribuição J2EE para poder ter suas funcionalidades básicas supridas.

O segundo ponto refere-se a estrutura hierárquica dos diretórios do Yale. A distribuição padrão do *framework* Yale obriga que a estrutura de diretórios em que esteja o arquivo principal (figura A.1), o yale.jar, siga um padrão pré-determinado para que a aplicação que o use funcione. Nesta estrutura encontram-se os diretórios com as outras bibliotecas da qual o Yale depende, os arquivos de configuração de usuários, arquivos de carregamento de módulos, dentre outros. O terceiro ponto a ser citado é o fato de que mesmo não sendo usado como ferramenta *stand-alone*, o Yale obriga o usuário a inicializar uma variável de ambiente do sistema operacional que indique o diretório raiz onde começa a sua hierarquia.

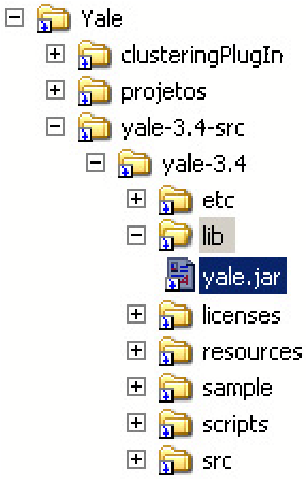

Figura A.1: Yale - Hierarquia de diretórios *Web*

Estas limitações tornaram-se um empecilho para uma adaptação simples do Yale ao *web container* Tomcat. O inconveniente causado pelo último ponto citado reside no fato

de que o administrador do sistema deve setar manualmente a variável durante a execução do Tomcat. Este empecilho é facilmente resolvido inserindo-se o comando Java *System.setProperty("yale.home","Caminho para o diretório raiz do Yale")* em um módulo de inicialização da ferramenta Ikasu. Este comando faz via código a inicialização da variável responsável por indicar o diretório raiz do Yale.

O primeiro e o segundo pontos se tornaram uma dificuldade devido ao esquema de funcionamento do uso de bibliotecas das aplicações rodando dentro do Tomcat. Para que uma aplicação execute de forma correta dentro do *container*, deve ter os arquivos de suas dependências colocados diretamente no diretório libs da aplicação.

Caso exista um diretório dentro do libs do Tomcat, este será ignorado. Isto implica em um problema com a hierarquia de diretórios imposta para o uso do Yale, pois já não se poderia ter um diretório raiz dentro desta pasta libs. No caso de colocarmos o arquivo principal do Yale e as outras bibliotecas de que depende, diretamente no diretório libs do Tomcat, a aplicação não funcionaria caso alguma função Yale fosse chamada. Isto ocorre porque durante sua execução o Yale procuraria os recursos que deveriam estar em sua hierarquia mas não encontraria os diretórios desejados. Existem duas soluções propostas para resolver estes problemas: adaptar a estrutura hierárquica do Yale à pasta libs no servidor ou mudar os fontes da biblioteca para que não use mais a estrutura antiga.

Na primeira solução, o arquivo principal do Yale seria colocado no diretório Libs do módulo e a hierarquia seria adaptada a esta posição. Nesta modelagem, o Yale seria chamado pelo Tomcat como uma biblioteca válida e usaria suas referências sem precisar ter seu código alterado. A desvantagem apresentada por esta abordagem está na ação de criar diretórios novos dentro da estrutura do módulo do Tomcat.

A segunda solução proposta para resolver o problema referente à pasta libs dos módulos do Tomcat e a hierarquia de diretórios rígida para o funcionamento do Yale, diz respeito a alterar o código fonte do *framework*. Esta alteração modificaria os diretórios em que o Yale procura suas dependências, para o mesmo diretório onde está instalado o arquivo principal. Assim não se precisaria criar novos diretórios dentro da estrutura de funcionamento do Tomcat. A desvantagem nesta solução está no fato de que a chance de erro na modificação de algum código dentro da distribuição padrão da biblioteca que provoque um comportamento indesejado é presente.