# DominoIA: Um Servidor de Jogos de Dominó

**Adailton de J. Cerqueira Jr.** [1,2] **, Adriano Veiga Botelho**[1,2] **, Fagner de A. M. Pimentel**[1,2]

[1]Núcleo de Arquitetura de Computadores e Sistemas Operacionais (ACSO)
Universidade do Estado da Bahia (UNEB) – Salvador – BA – Brasil

[2]Departamento de Ciências Exatas e da Terra - DCET-I
Universidade do Estado da Bahia (UNEB) – Salvador – BA – Brasil

{adailton.junior, profpardal.88, fagnerpimentel}@gmail.com

***Abstract.*** *This paper describes the development of a domino game application for autonomous intelligent agents. Using computer games concepts, we built the software DominoIA, which have as objective to provide an environment to assist the development of Artificial Intelligence for domino player agents, through a simple comunication protocol and through an exhibition of matches in a graphical interface. We'll show here too the DominoIA's arquitecture, also our caution during development and the results.*

***Resumo.*** *Este artigo descreve o desenvolvimento de um servidor de jogos de dominó entre agentes inteligentes autônomos. Utilizando conceitos de jogos computacionais, construímos o software DominoIA, que tem como objetivo prover um ambiente para auxiliar o desenvolvimento da Inteligência Artificial para agentes jogadores de dominó, através de um protocolo de comunicação simples e da exibição dos jogos em uma interface gráfica. Mostraremos também a arquitetura utilizada, bem como os cuidados durante seu desenvolvimento e os resultados obtidos.*

## 1. Introdução

O projeto do servidor de jogos de dominó surgiu na matéria de Inteligência Artificial (IA) do curso de Sistemas de Informação da UNEB, campus I. Foi sugerido como avaliação uma competição entre agentes inteligentes que fossem capazes de jogar dominó sozinhos, desenvolvidos por diferentes equipes de alunos.

Mas para avaliar o desempenho dos mesmos e poder compará-los, era necessário um servidor que se comunicasse com os agentes e gerenciasse as partidas de dominó, seguindo as regras definidas no campeonato proposto. Uma parte dos integrantes de cada equipe formariam então um novo grupo, que seria responsável por criar este servidor, em que os agentes pudessem atuar nele.

Surge então o DominoIA, que será descrito com mais detalhes neste artigo. A seção 2 apresenta um breve histórico e alguns conceitos e características dos jogos e a presença da IA nos mesmos. Na seção 3 faremos uma breve análise sobre alguns trabalhos relacionados. Logo após, na seção 4, discutiremos a arquitetura do software, apresentando seus principais módulos na seção 4.1 e o protocolo utilizado para a comunicação com os agentes na seção 4.2. A seção 5 relata como se deu o desenvolvimento do software e a seção 6 mostra os testes realizados e os seus resultados. Por fim, a seção 7 faz uma conclusão e apresenta os nossos trabalhos futuros.

## 2. Jogos e IA

Segundo [Amorim 2002], os jogos podem ser divididos em dois grupos: aqueles dependentes do mundo físico e os independentes do mundo físico. No primeiro grupo encontra-se os jogos onde a prática esta intrinsecamente ligada às leis fisicas, por exemplo, futebol, tênis, vôlei, etc. No segundo grupo, temos os jogos nos quais o suporte físico não é a essência do jogo, por exemplo, o jogo de damas independe da forma que utilizaremos para representar as peças, podemos utilizar desde pedras coloridas até "pedacinhos"de papel. Neste grupo, estão os jogos de cartas e de tabuleiro.

Os jogos dependentes do mundo físico ou jogos físicos possuem muitas variáveis de ações possíveis e regras bastante imprecisas, que normalmente necessitam de um juiz para legalizar a jogada. Por estes motivos, os jogos físicos não atraíram o interesse dos pesquisadores na área de inteligência artificial [Russel and Norvig 2002], com raras exceções, a exemplo do futebol de robôs [Kitano et al. 1997].

Os jogos independentes do mundo físico ou jogos abstratos, ao contrário, possuem um pequeno número de ações e regras bastante precisas. Apesar disto, os jogos abstratos são interessantes, pois possuem soluções complexas. Por exemplo, o xadrez tem uma média de ramificação de 35 e as partidas chegam até a 100 movimentos; com isso temos uma árvore de busca com $35^{100}$ nós [Russel and Norvig 2002].

Os jogos de azar, como a roleta e os dados não são considerados jogos de verdade, sob uma perspectiva computacional, pois os jogadores não exercitam nenhuma influência no jogo, deixando somente o caráter de aleatóriedade influenciar no resultado.

### 2.1. Características dos Jogos

Em [Amorim 2002], um jogo caracteriza-se por:

1. Um espaço de estados: É o conjunto de arranjos possíveis das peças.
2. Um estado inicial: É o conjunto inicial de arranjos das peças.
3. Um conjunto de operadores: Determinam a passagem entre os estados.
4. Um estado final: É o conjunto de arranjos das peças que caracteriza uma vitória ou um empate.

Todos os arranjos das peças devem ser possíveis pela definição da regra do jogo.

Os jogos podem ser classificados de acordo com os elementos que os jogadores possuem para tomar as suas decisões. Podendo ser de informação completa, em que os jogadores tem acesso completo à disposição das peças ou de informação incompleta, além do acesso parcial à disposição das peças.

Outra classificação dos jogos leva em consideração o grau de aleatoriedade. Jogos que não envolvem a sorte, na transição de um estado para outro, são chamados de jogos não-probabilísticos. Enquanto os jogos que envolvem sorte a cada nova jogada, além das decisões dos jogadores, são chamados de jogos probabilísticos.

O dominó pode ser classificado de três formas, de acordo com as variadas modalidades de jogo. Quando há dois jogadores com quatorze peças cada, o jogo se classifica como não-probabilístico e de informação completa, pois não existem peças de cabeça para baixo, para serem apanhadas de forma aleatória e basta cada jogador contar as peças

**Tabela 1. Tabelas de Classificação dos Jogos.**

| | **Não-Probabilístico** | **Probabilístico** |
|---|---|---|
| **Informação Completa** | Xadrez<br>Go<br>Dominó [a] | Gamão<br>WAR<br>Banco Imobiliário |
| **Informação Incompleta** | Dominó [b] | Pôquer<br>Dominó [c] |

[a]Duas pessoas com quatorze peças cada.
[b]Em duplas.
[c]Duas pessoas com sete peças cada.

da sua mão e da mesa para saber as peças da mão do oponente. No jogo em duplas, é classificado como não-probabilístico e de informação incompleta, já que não existem pedras para serem apanhadas aleatoriamente, mas como as peças estão distribuídas entre os quatro jogadores, nenhum deles consegue ter acesso à combinação exata de peças dos demais. Entretanto, no decorrer do jogo, ele tende a um jogo de informação completa, já que mais pedras estarão na mesa de cabeça para cima, diminuindo, assim, a combinação de peças nas mãos dos oponentes. Por último, temos dois jogadores com sete pedras cada, este tipo de jogo é classificado como probabilístico e de informação incompleta, pois nenhum dos jogadores possui a combinação exata da mão do oponente e o elemento de aleatoriedade torna-se parte do jogo, já que temos peças para serem apanhadas.

Devemos também distiguir os jogos dos quebra-cabeças. Nos jogos existem dois ou mais jogadores competindo entre si. Enquanto nos quebra-cabeças o desafio não envolve o enfrentamento de adversários e sim, a solução de um problema. Ainda que se utilize algum critério de competição, como tempo ou menos movimentos para solução, não existe a presença de adversários atrapalhando o jogador. Deve-se admitir que os quebra-cabeças são um bom campo de desafio para inteligência artificial [Amorim 2002]. Entretanto o enfrentamento nos jogos torna o desafio mais lúdico e coloca os programas em uma prova desempenho mais forte e atraente.

## 3. Trabalhos Relacionados

Esta seção visa abordar trabalhos relacionados na área de jogos e desenvolvimento de IA, mostrando o funcionamento e objetivo destes projetos. Além de suas características similares ao DominoIA no incentivo ao ensino da programação e desenvolvimento de IA.

### 3.1. Odin

Esta é uma ferramenta que permite construir especificações executáveis de teorias de IA aplicada a jogos. Pode ser usada em disciplinas de IA para o desenvolvimento de projetos didáticos estimulantes e atraentes para os alunos, que podem apreciar os resultados de seus trabalhos como protótipos de jogos por eles desenvolvidos [da Silva et al. ].

### 3.2. Soar/Games

O objetivo do projeto Soar/Games é aplicar técnicas do estado da arte de IA a jogos de computadores por meio do desenvolvimento de agentes inteligentes [Jenkins 2001].

A engine do Soar/Games é dividida em três partes: (1) Máquina de inferência que aplica o conhecimento do agente à situação atual, operando em um ciclo de decisão: perceber, pensar e agir, (2) base de conhecimento composta por objetivos e táticas, e (3)interface com o jogo que serve de canal de comunicação da máquina de inferência com o ambiente real.

### 3.3. Muppets

Muppets (Multi User Programming Pedagogy for Enhancing Traditional Study) [Phelps et al. 2003] tem como objetivo atenuar a difícil experiência dos calouros em cursos voltados para a programação de computadores [da Silva et al. ]

O Muppets apresenta três grupos de interface: (1) interface tridimensional para visualização do ambiente, (2) console e (3) IDE(Integrated Development Environment) que pode ser usado dentro do ambiente 3D [Bierre and Phelps 2004].

### 3.4. Jogos de Dominó

Outros trabalhos relacionados são os diversos jogos de dominó que podem ser encontrados disponíveis para download na web, como o Dominó plus 2.0 [Plus ] e o Dominó master 3.0 [Master ]. Estes são softwares que além de prover um ambiente de jogo assim como o DominoIA, também fornecem adversários providos de IA para enfrentar adversários humanos.

## 4. Arquitetura de Software do DominoIA

O software DominoIA foi definido em duas partes principais: um módulo central conhecido como Servidor e outro módulo chamado de Gráfico, vistos na figura 1. O módulo servidor é responsável pela condução correta do jogo, conexão e transmissão das mensagens dos clientes, aplicação das regras de dominó e geração de um log que serve para exibição do jogo pelo módulo gráfico. Por sua vez, o módulo gráfico é responsável pela leitura do log gerado pelo servidor e apresentação dos jogos ocorridos no servidor de forma lúdica.

O que motivou a separação do módulo gráfico do módulo servidor é que as partidas seriam muito rápidas para serem acompanhadas por seres humanos. Com isso, várias partidas poderiam ser rodadas em paralelo, não havendo a necessidade de serem acompanhadas em tempo real.

Outro motivo é que um baixo acoplamento entre os módulos permite inclusive que os mesmos sejam desenvolvidos e aperfeiçoados de forma independente, só havendo a necessidade de se seguir o protocolo de comunicação entre eles.

### 4.1. Módulos do DominoIA

O servidor é dividido em sub-módulos, onde cada um é responsável por cada parte do servidor, são eles:

- Cliente
- Juiz
- Passer
- Server

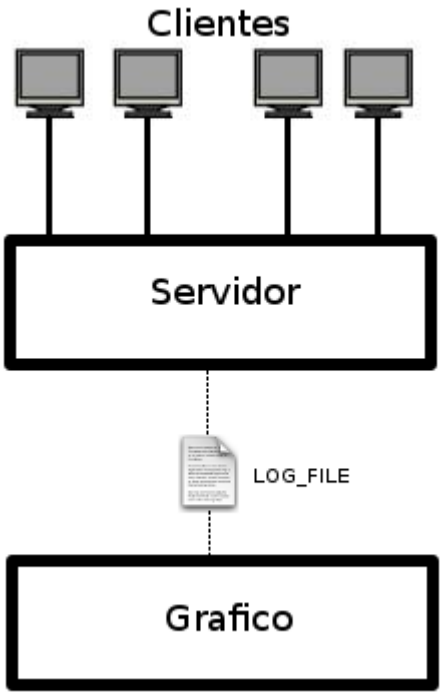


**Figura 1. Arquitetura modular do Servidor DominoIA**

- XML

No sub-módulo Cliente são processadas e armazenadas as informações referentes aos clientes. Como por exemplo, o armazenamento do nome do jogador e do time, a composição das pedras na mão, as jogadas do cliente, a pontuação, a transmissão de mensagens, entre outras funções.

O Juiz é responsável pela aplicação das regras do jogo de dominó: contagem de tempo, verificação de final das partidas, gerenciamento da pontuação e tipo de batida.

O sub-módulo Server é responsável pela construção do jogo e atualizações no mundo. Ações como a distribuição das pedras, construção da mesa e fluxo do jogo encontram-se neste módulo. Este também é o módulo central do Servidor, onde todos os demais são instanciados.

O XML é responsável pela construção de todas as mensagens enviadas pelo servidor para os clientes, assim como a geração de log, carregado pelo módulo gráfico no final do jogo. No Parser é feita a recepção, interpretação, validação e transmissão das mensagens do cliente para os demais sub-módulos do servidor.

O módulo gráfico está divido em dois sub-módulos principais: o Graphics e o ReplayMode. O primeiro está relacionado com toda a parte de geração, carregamento e exibição de imagens e textos. O segundo está relacionado com a leitura do arquivo de log XML gerado pelo servidor e atualização dos dados. O mesmo também é responsável pela interação com o usuário, permitindo-o navegar entre as jogadas, avançando e retrocedendo.

Desta forma, a exibição de imagens e a leitura do arquivo gerado pelo servidor se dá de forma independente. Isto é interessante pois nos permitirá, no futuro, adicionar novas funcionalidades sem precisar alterar a exibição gráfica na tela. Por exemplo, podemos criar um novo sub-módulo que se conecte ao servidor e atualize seus dados em tempo real. Assim, basta utilizar a mesma interface do sub-módulo Graphics e o jogo será exibido na tela.

Toda a parte gráfica foi desenvolvida através da Allegro [Allegro ], uma biblioteca multiplataforma de código aberto utilizada para a programação de jogos e de aplicativos multimídia em geral. Ela é escrita em C/C++ e possui funções que facilitam bastante o trabalho com imagens, sons, temporizadores e a entrada de dados via teclado, mouse ou joystick.


**Figura 2. Tela do Gráfico.**

### 4.2. O protocolo de comunicação XML

Os protocolos utilizados no servidor definem a forma com que o servidor se comunica com os clientes e o módulo gráfico. Para isso foram definidos três tipos de protocolos:

- Servidor-Cliente
- Cliente-Servidor
- Servidor-Gráfio

Estes protocolos foram construídos com a *Extensible Markup Language* (XML) proporcionando uma padronização nas mensagens entre os módulos e os clientes.

Os protocolos Servidor-Cliente e Cliente-Servidor são divididos em pacotes de mensagens que são enviadas conforme o fluxo do jogo, enquanto o Servidor-Gráfico é montado completamente e enviado ao final do jogo em forma de arquivo de texto.

O protocolo Servidor-Cliente se divide em:

- Conexão - onde se confirma os nomes e IDs dos clientes que tentaram se conectar ao servidor;
- Init_Jogo - que envia para todos os clientes quem são seus adversários e companheiro;
- Envio_de_Mão - onde envia para cada cliente suas pedras sorteadas pelo servidor;
- Solicitação_de_Vontade - onde o servidor solicita a vontade que um jogador tem de iniciar o jogo (quando esse jogador faz parte do time que bateu sem fechar uma partida ou uma peça na última rodada);
- Init_Rodada - identifica a partida atual, o placar atual e o jogador que começa a rodada;
- Loop - identifica as jogadas feitas pelos clientes no jogo com o ID do cliente que fez a jogada, a peça e a ponta que o cliente jogou;
- Fim_Rodada - envia a duração da rodada, o tipo de batida e o time vencedor;
- Fim_Partida - onde se envia a duração total e pontuação geral da partida.

O protocolo Cliente-Servidor se divide em:

- Conexão - onde se solicita a conexão com o servidor;
- Init - onde se envia a vontade de se iniciar o jogo quando solicitado pelo servidor;
- Loop - onde o cliente envia sua jogada com seu ID, ponta e pedra que deseja jogar.

O fluxo de transmissão de cada mensagem do protocolo pode ser visto no diagrama da figura 3. O protocolo Servidor-Gráfico, como dito antes, é enviado por completo como um arquivo de texto, entretanto, ele é construído por partes no servidor. Este protocolo é dividido em “init ”, onde são passadas as informações dos clientes e dos times participantes do jogo; e “partida ”, que contém a data, horário e as rodadas de cada partida. Por sua vez, as rodadas são divididas em “init rodada ”, onde são transmitidas as pedras e vontades dos clientes; “loop rodada ”, onde identifica o número da jogada, quem jogou, a ponta e a peça jogada; e “fim rodada ”, onde se envia a duração da rodada, quem bateu, o tipo de batida e a pontuação do jogo.

## 5. O Desenvolvimento do DominoIA

Após a definição dos módulos e sub-módulos, foi dado início à fase de desenvolvimento do servidor. A equipe de desenvolvimento, que contava com cinco integrantes, foi dividida em duas sub-equipes. Uma equipe responsável pelo desenvolvimento do módulo do Servidor e outra para o módulo do Gráfico.

O objetivo principal do desenvolvimento do servidor foi a construção de um código modularizado que facilitasse o acréscimo de novas funcionalidades, além da preocupação em seguir boas práticas de programação, como um código claro, uma boa identação e ausência de números “mágicos”.

Também houve a preocupação em evitar as causas do envelhecimento de código, tais como o não cumprimento das especificações originais de projeto, requisitos imprecisos ou incompletos, prazos curtos, ferramentas de desenvolvimento inadequadas, falta de documentação, equipes desniveladas ou com alta rotatividade, que foram sempre minimizadas, para evitar um envelhecimento precoce do software [Eick et al. 2001].

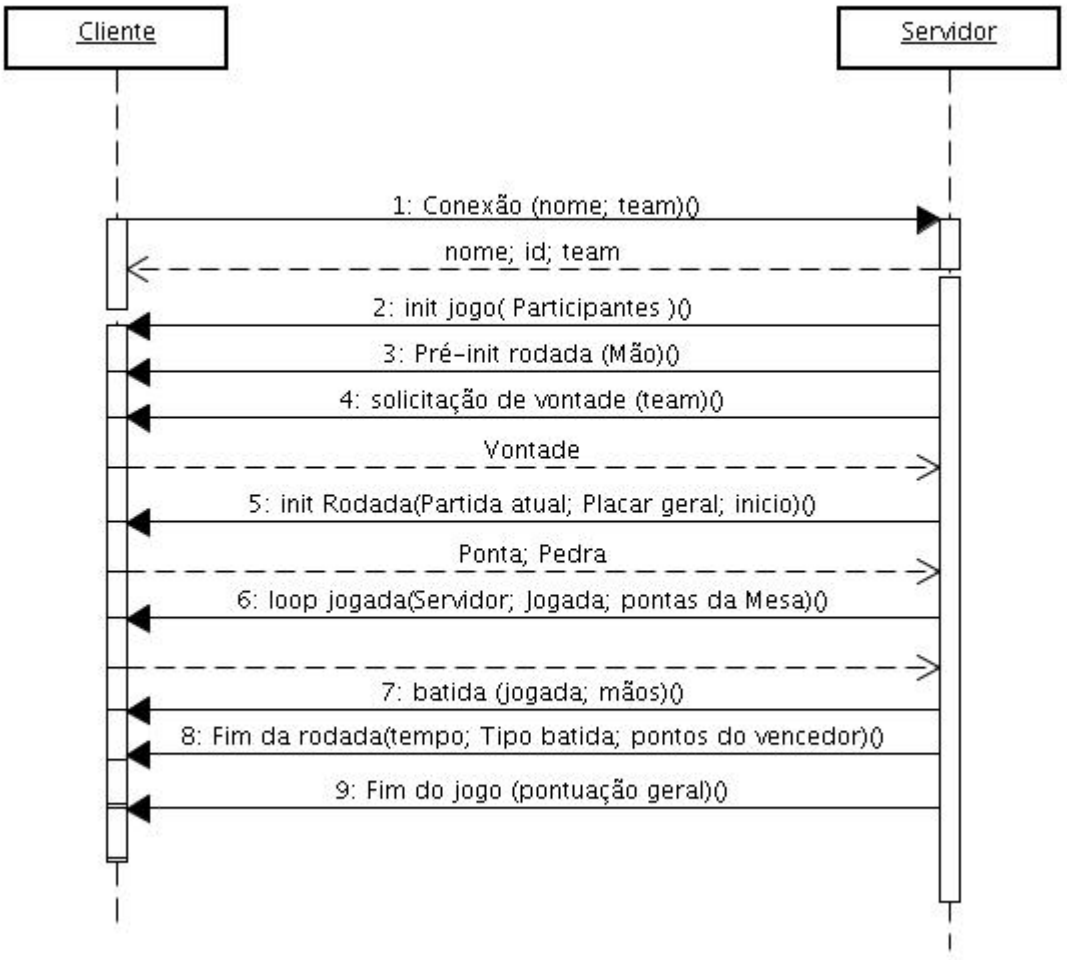


**Figura 3. Diagrama de sequência do Software DominoIA.**

Desenvolvido na linguagem C/C++ e inicialmente desenvolvido para executar no Sistema Operacional Ubuntu 8.10, mas com o código totalmente portável para outras distribuições. O motivo é que o servidor utiliza somente bibliotecas padrões do C/C++ ou bibliotecas multiplataformas, como a Boost libraries [Boost ].

## 6. Testes e Resultados

O servidor foi testado com base na técnica de teste da caixa-preta ou teste funcional. Esta técnica avalia a funcionalidade externa do software, analisando apenas as entradas e saídas do programa [Myers 2004].

Para auxiliar na depuração do Servidor foi utilizado o Valgrind. O Valgrind é uma ferramenta software livre que detecta erros decorrentes do uso incorreto da memória dinâmica, como por exemplo os vazamentos de memória, alocação e desalocação incorretas e acessos em áreas inválidas [Valgrind ].

Por fim, o principal teste do DominoIA foi na apresentação dos agentes inteligentes jogadores de dominó da matéria de Inteligência Artificial do curso de Sistemas de Informação da UNEB, campus I. Ainda em sua versão beta, as partidas entre os agentes foram executadas com sucesso.

## 7. Conclusões e Trabalhos Futuros

Este artigo apresentou o software DominoIA, suas funcionalidades, funcionamento e comunicação na construção de uma plataforma para desenvolvimento de inteligência artificial utilizando conceitos de jogos computacionais. Com este software foi possível realizar jogos em que clientes autônomos jogadores de dominó executavam jogadas básicas com um mínimo de IA, dando assim os primeiros passos para um desenvolvimento mais complexo.

Como trabalho futuro, pretendemos continuar o desenvolvimento do servidor, implementado novas funcionalidades e novos jogos utilizando o dominó, aumentado a complexidade tanto no jogo como no uso das técnicas de IA. Este ano ainda serão realizados outros campeonatos, a exemplo dos que ocorrerão no InfoUNEB, em que os participantes, na sua maioria iniciantes no curso, poderão se interessar no desenvolvimento de IA, ajudando assim a difundir os estudos na área, além do desenvolvimento de um módulo onde jogadores humanos possam desafiar os jogadores autônomos.

## Referências


Allegro. Allegro c++. http://www.allegro.cc/. Acessado em 14 de dezembro de 2009.

Amorim, C. A. d. (2002). A maquina e seus limites: Uma investigação sobre o xadrez computacional. Master's thesis, Universidade Federal da Bahia.

Bierre, K. and Phelps, A. (2004). The use of muppets in an introductory java programming course. In *Proceedings of the 5th conference on Information technology education*, pages 122–127. ACM.

Boost. Boost c++ libraries. http://www.boost.org/. Acessado em 03 de novembro de 2009.

da Silva, F., em Ciências, M., de Concentração, A., and da Computação, C. Uma ferramenta para o ensino de inteligência artificial usando jogos de computador.

Eick, S., Graves, T., Karr, A., Marron, J., and Mockus, A. (2001). Does code decay? assessing the evidence from change management data. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, pages 1–12.

Jenkins, T. (2001). The motivation of students of programming. *Proceedings of the 6th annual conference on Innovation and technology in computer science education*, pages 53–56.

Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., and Osawa, E. (1997). Robocup: The robot world cup initiative. In *AGENTS '97: Proceedings of the first international conference on Autonomous agents*.

Master, D. Dominó master. http://www.baixaki.com.br/download/domino-master.htm. Acessado em 31 de março de 2010.

Myers, G. J. (2004). *The art of software testing*. John Wiley & Sons, New York.

Phelps, A., Bierre, K., and Parks, D. (2003). Muppets: multi-user programming pedagogy for enhancing traditional study. In *Proceedings of the 4th conference on Information technology curriculum*, page 105. ACM.

Plus, D. Dominó plus. http://www.baixaki.com.br/site/dwnld16026.htm. Acessado em 31 de março de 2010.

Russel, S. and Norvig, P. (2002). *Artificial Intelligence*. Prentice-Hall, 2 edition.

Valgrind. Valgrind. http://valgrind.org/. Acessado em 28 de janeiro de 2010.