Pesquisa e mapeamento de Shells, Frameworks e jogos para auxiliar no ensino de Inteligência Artificial

Heverton de Lemos[1], Anita Maria da Rocha Fernandes[1]
[1]Universidade do Vale do Itajaí (UNIVALI)
São José – SC – Brasil
hevertoncn@gmail.com, anita.fernandes@univali.br

Abstract. *Artificial Intelligence (AI) is a subject taught in the final semesters of courses in Computer Science and its content is very extensive and complex, so often the teacher chooses to transform the discipline in an informative matter. To occur at the end of the course and be very theoretical, this discipline becomes uninteresting to the students, since they are more concerned with the work completion. The use of Shells, Frameworks, and games related to AI techniques are ways to encourage the practice of concepts and motivating lessons, so this paper presents a mapping of such tools to assist in the teaching/ learning process AI.*

Resumo. *Inteligência Artificial (IA) é uma disciplina ministrada nos semestres finais dos cursos de Ciência da Computação e seu conteúdo é muito extenso e complexo, por isso muitas vezes o professor opta por transformar a disciplina em uma matéria informativa. Por ocorrer no final do curso e ser muito teórica, esta disciplina se torna desinteressante para os alunos, visto que eles estão mais preocupados com os trabalhos de conclusão de curso. O uso de Shells, Frameworks, e jogos relacionados às técnicas de IA são formas de estimular a prática dos conceitos e motivar as aulas, sendo assim, este artigo apresenta um mapeamento de tais ferramentas para auxiliar no processo de ensino/aprendizagem de IA.*

1 Introdução

A queda do interesse por parte dos universitários, para cursos de Ciência da Computação e afins em várias universidades do mundo [Perelman 2013; Vegso 2013], é um sinal para que os professores desta área passem a pensar se um dos motivos dessa queda tem relação com a forma pela qual o ensino de computação está sendo conduzido. Nesta linha, pergunta-se se existem maneiras de tornar o ensino de computação mais interativo e motivante para os alunos da nova geração, os quais cresceram no meio de um mundo de recursos computacionais.

Dentro deste contexto, disciplinas como a de Inteligência Artificial (IA) precisam ter suas formas de ensino revistas com o objetivo de cativar os alunos. Ramalho (2012) é um forte crítico do ensino de Inteligência Artificial no Brasil. Segundo ele, existem três problemas que norteiam o desinteresse pela disciplina. Primeiro, nas grandes universidades brasileiras, são ofertadas poucas matérias relacionadas ao campo da Inteligência Artificial nos cursos da área de computação. Das matérias ofertadas, menos ainda são obrigatórias, o que acaba fazendo com que a Inteligência Artificial passe despercebida por muitos currículos. Além disto, o fato de que as disciplinas de Inteligência Artificial são ofertadas mais tarde, ou seja, na segunda

metade do curso, acaba por afastar as mesmas das disciplinas consideradas de base, e isto faz com que o aluno não ache conexão entre o que foi aprendido no começo do curso com a disciplina de Inteligência Artificial. Por fim, o autor aponta que as ementas são muitas vezes restritas e/ou desatualizadas se comparadas com as grades universidades como o MIT (*Massachussets Institute of Technology*). Muito foco ainda é dado para as abordagens excessivamente teóricas sobre busca e representação do conhecimento, além de que em muitos casos alguns currículos buscam apresentar de tudo um pouco dentro da vasta área que é a Inteligência Artificial, porém, como na maioria das vezes, há uma única disciplina deste conteúdo no curso, os alunos acabam conhecendo superficialmente alguns conceitos sem se aprofundar muito. Nesta configuração, solicitar trabalhos de implementação das técnicas torna-se inviável.

Como consequência, as aulas em sua maioria acabam se tornando monótonas. Sendo assim, há uma tendência em se buscar aplicar recursos como jogos, *Shells* e *Frameworks* que auxiliem no aprendizado dos conceitos e também na implementação das técnicas de IA. Dentro deste contexto, este artigo apresenta a pesquisa e mapeamento de jogos, *Shells* e *Frameworks* que possam auxiliar no ensino dos conteúdos de Inteligência Artificial.

1.1 Problema

A Inteligência Artificial é uma área de pesquisa recente, apesar disto, Coppin (2004) aborda o fato de suas bases surgirem há milhares de anos e suas influências serem dos mais variados contextos. De maneira geral, Tanimoto (1987) conceitua Inteligência Artificial como um campo de estudos que engloba uma série de técnicas computacionais, com o intuito de executar tarefas que requerem inteligência e conhecimento humano de maneira eficiente.

De acordo com a Sociedade Brasileira de Computação [Sbc 2013], os tópicos que devem ser abordados em uma disciplina de Inteligência Artificial são: Linguagens Simbólicas, Programação em Lógica, Resolução de Problemas como Busca, Estratégias de Busca e Busca Heurística, Representação do conhecimento, Sistemas baseados em conhecimento, Lógica *Fuzzy*, Aprendizado de máquina, Redes Neurais, Algoritmos Evolutivos, Processamento de Linguagem Natural, Agentes Inteligentes.

Dentro desta grande quantidade de conteúdos, o professor tem um semestre para transmitir os conceitos aos alunos da disciplina e muitas vezes este processo acaba sendo meramente teórico. Solicitar trabalhos de implementação nos quais o aluno tenha que partir do zero demanda um tempo que a disciplina não dispõe. Além disto, para uma disciplina de final de curso, conteúdos meramente teóricos se tornam desmotivantes. Sendo assim, a busca por ferramentas que tornem a disciplina mais atraente e motivadora é essencial.

Este artigo apresenta nas próximas seções a pesquisa inicial feita junto a alunos e professores, a fim de levantar as ferramentas mais usadas por eles. Em seguida será apresentado o levantamento curricular feito em algumas universidades, com o intuito de verificar quais os conteúdos mais comuns às disciplinas de IA, buscando focar as necessidades dos mesmos. Por fim são apresentadas as principais ferramentas pesquisadas e o portal desenvolvido com as informações referentes ao artigo focando nas ferramentas citadas neste, para facilitar pesquisas sobre tal conteúdo.

2 Pesquisa de Campo

Para realizar o levantamento de *Shell*s, *Frame*works e jogos utilizados no auxílio do ensino/aprendizado da matéria de Inteligência Artificial, inicialmente realizou-se uma pesquisa sobre ferramentas disponíveis que podem ser usadas nesta disciplina, escolhendo-se algumas arbitrariamente e, então, foi elaborado um questionário *online* com perguntas simples e objetivas. Participaram da pesquisa alunos, ex-alunos e professores da UNIVALI, bem como alunos e professores de outras instituições, num total de 31 participantes. Os resultados estão apresentados na Figura 1.

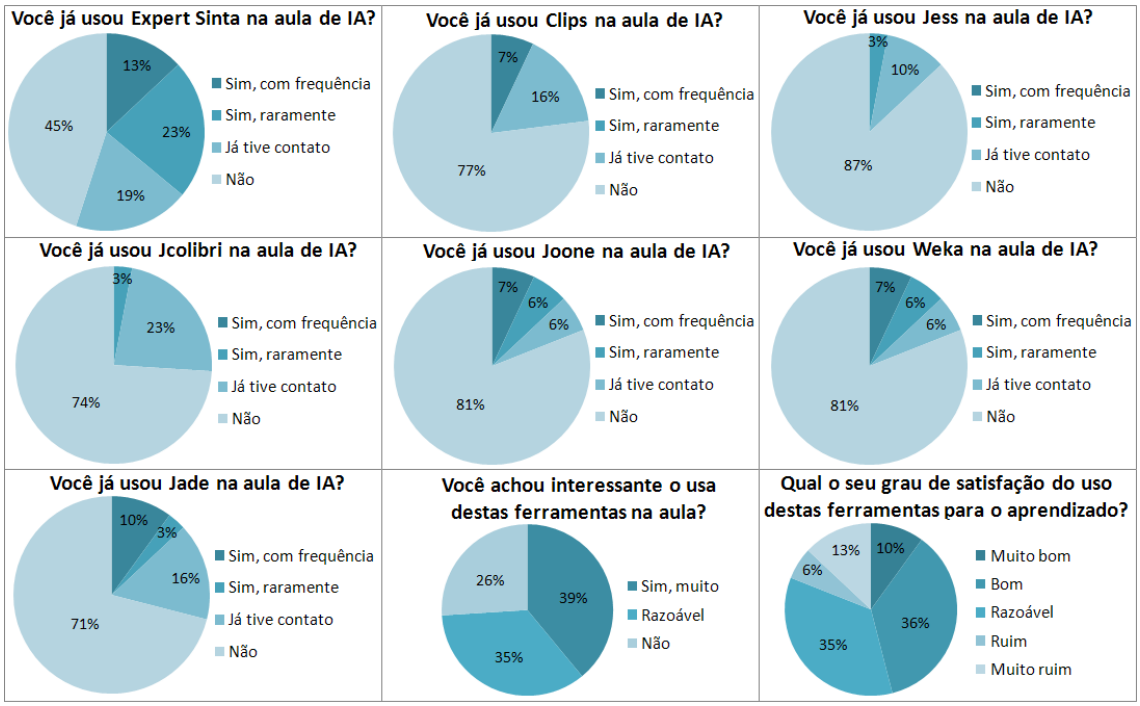

Figura 1 – Gráficos referentes à pesquisa de campo realizada.

3 Análise de Currículos

Para a realização desta etapa do projeto, foi realizada uma busca por ementas da disciplina "Inteligência Artificial" nas universidades que oferecem cursos na área da computação, como Ciência da Computação e Sistemas da Informação.

Feito isso, tais ementas foram analisadas e comparadas entre si, a fim de se obter uma relação dos conteúdos mais abordados. Após a análise, constatou-se que a maioria dos cursos aborda os seguintes temas: Introdução a IA; Teoria de Problemas; Paradigma Simbólico da IA; Modelagem de Agentes Inteligentes; Métodos de Busca; Sistemas Baseados em conhecimento; Métodos de Inferência; Tratamento de Incertezas; Sistemas Especialistas; Lógica Nebulosa/*Fuzzy*; Paradigma Conexionista da IA/redes neurais; Paradigma Evolucionário da IA; *Data Mining*; IA em Jogos; Processamento de Linguagem natural; e Aprendizado de máquina.

A partir deste conteúdo e considerando o resultado da enquete feita junto a alunos e professores de IA, a próxima etapa do projeto foi mapear as ferramentas.

4 Ferramentas

As ferramentas mapeadas focaram em *Frameworks, Shells* e jogos.

4.1 Frameworks

Para Fayad *et al* (1999a), um *Frame*work é um conjunto de classes que constitui um projeto abstrato para a solução de uma família de problemas. Para Mattsson (1996 e 2000), é uma arquitetura desenvolvida com o objetivo de atingir a máxima reutilização, representada como um conjunto de classes abstratas e concretas, com grande potencial de especialização. Para Gamma *et al.* (1995), é um conjunto de objetos que colaboram com o objetivo de atender a um conjunto de responsabilidades para uma aplicação específica ou um domínio de aplicação. Já segundo Buschmann *et al*. (1996) e Pree (1885), é definido como um *software* parcialmente completo projetado para ser instanciado. Os *Frameworks* podem ser classificados de diversas formas. Inicialmente, são classificados em dois grupos principais: *Frameworks* de aplicação orientados a objetos [Fayad 1999a; 1999b; 2000] e *Framework* de componentes [Szyperski 1997].

A seguir serão apresentados os principais *Frameworks* pesquisados.

4.1.1 Joone

O "*Java Object Oriented Neural Engine"* (Joone) é um *Framework* livre utilizado para construir e executar aplicativos de Inteligência Artificial (IA) baseados em redes neurais. Ele consiste em uma arquitetura modular baseada em componentes conectáveis que podem ser estendidos para a construção de novos algoritmos de aprendizagem e arquiteturas de redes neurais [Marrone 2004]. Criado por Paolo Marrone e sua equipe, este *Framework* é escrito 100% em Java, e pode ser executado em qualquer plataforma para a qual a Java Runtime Environment está disponível.

Sua primeira versão foi lançada em 2004 e a última (2.0) em 2007. E hoje, em 2013, o mesmo tem sido pouco utilizado. Seu guia completo, escrito por seu próprio autor, é considerado seu principal documento [Marrone 2004].

4.1.2 Jcolibri

O JCOLIBRI é um *Framework* orientado a objetos que ajuda a projetar sistemas de Raciocínio Baseado em Casos (RBC). Desenvolvido pelo GAIA (*Group for Artificial Intelligence Applications*), este teve sua primeira versão disponibilizada, em 2005, em um ambiente não muito usual para a elaboração de sistema, e sua segunda versão não possui interface gráfica para elaboração e testes do sistema RBC [Gaia 2013; Fernandes 2012]. Este é um *Framework* portável e robusto, que não possui uma interface para elaboração dos sistemas de RBC e que disponibiliza poucas métricas de similaridade. Ele é um *software* livre desenvolvido com a linguagem de programação Java e possui uma arquitetura projetada para ser extensível e reutilizável em diferentes domínios e famílias de RBC [Gaia 2013; Fernandes 2012].

O JCOLIBRI teve somente dois grandes lançamentos: versões 1 e 2. Na primeira versão (lançada em 2005) ele inclui uma interface gráfica completa, que orienta o usuário na concepção de um sistema CBR. Esta versão é recomendada para usuários não desenvolvedores que desejam criar sistemas CBR sem programar qualquer código. A

segunda versão (2008) apresenta uma nova aplicação que segue uma nova e clara arquitetura dividida em duas camadas: uma voltada para desenvolvedores (terminado) e outra voltada para *designers* (trabalho futuro) [Gaia 2013].

4.1.3 Weka

O "*Waikato Environment for Knowledge Analysis*" (Weka) é uma das ferramentas mais completas em mineração de dados. Como uma plataforma de mineração de dados pública, o Weka reúne algoritmos de aprendizado para explorar dados, incluindo o pré-tratamento dos mesmos, classificação, regressão, mineração de regras de associação e visualização de nova interface [Zhong 2011; Waikato 2013].

O Weka foi projetado para que se possa experimentar rapidamente os métodos existentes em novos conjuntos de dados de formas flexíveis. Ele fornece suporte extensivo para todo o processo de mineração de dados experimental, incluindo a preparação dos dados de entrada, avaliação de sistemas de aprendizagem estatística e o resultado de aprendizagem. Além de uma grande variedade de algoritmos de aprendizagem, que inclui uma vasta gama de ferramentas de pré-processamento. Este conjunto de ferramentas diversificada e abrangente é acessado através de uma interface comum para que seus usuários possam comparar diferentes métodos e identificar aqueles que são mais adequadas para o problema em questão [Witten 2005].

A primeira versão do Weka foi somente interna e ocorreu em 1994. E a primeira versão pública (na versão 2.1) foi feita em 1996. Sua versão mais recente (Weka 3), começou a ser desenvolvida em 1997 e hoje é utilizada em aplicações de diferentes áreas, especialmente, para fins educacionais e de pesquisa. Já a versão 3.7.10 está em desenvolvimento neste ano de 2013 [Frank *et al.*; Waikato 2013].

4.1.4 Jade

O "*Java Agent Development Framework*" (JADE) é um *Framework* de desenvolvimento totalmente implementado em Java. Embora o JADE forneça todos os componentes obrigatórios (*Foundation for Intelligent Physical Agents* - FIPA) para o desenvolvimento de agentes autônomos, falta-lhe a capacidade de incluir comportamento inteligente para os agentes individuais [Balachandran 2008; Jade 2013]. Os primeiros desenvolvimentos do *software*, que se tornou a plataforma JADE, foram iniciados pela Telecom Italia (anteriormente CSELT) em 1998, e a sua última versão, a 4.3.0, foi lançada em março de 2013. O "*White paper*", seu documento, dá uma visão geral da plataforma JADE, apresenta sua arquitetura e suas principais funcionalidades [Jade 2013; Bellidemine *et al.* 2007; Caire 2003].

O JADE é um *software* livre e é distribuído pela Telecom Italia, detentora dos direitos autorais, em *software* de código aberto sob os termos do *Lesser General Public License Version* 2 (LGPL). Esta licença garante todos os direitos básicos para facilitar o uso do *software* incluído em produtos comerciais: o direito de fazer cópias do *software* e distribuí-las, de ter acesso ao código-fonte e de alterar o código e fazer melhorias [Jade 2013; Bellifemine *et al.* 2007].

4.2 Shells

De acordo com Eugênio e Palermo (2000), *Shell* é um programa que conecta e interpreta os comandos que o usuário digita. Pode ser considerado como uma linguagem de programação completa, possuindo variáveis, construções condicionais, interativas e ambiente adaptável ao usuário. Permite ao usuário realizar suas atividades sem afetar qualquer outro processo que não lhe pertence. Pode-se considerar *Shell* como sendo uma das linguagens de quarta geração.

Para Maia (2012), dentro do contexto da Inteligência Artificial, *Shell* é uma ferramenta computacional projetada para o desenvolvimento de sistemas baseados em conhecimento, apenas com a inclusão de uma base de conhecimento. Um exemplo clássico é o EMYCIN, um *Shell* para desenvolvimento de sistemas especialistas criado a partir da remoção da base de conhecimento médico do sistema especialista MYCIN. No início das pesquisas em Inteligência Artificial predominavam os *Shells* focados em Sistemas Especialistas, porém logo as demais técnicas foram sendo contempladas com *Shells* que dessem suporte ao seu desenvolvimento. *Shells* tanto comerciais quanto a nível acadêmico.

Um *Shell* refere-se a um programa que permite ao usuário interagir com o computador através de algum tipo de interface. Normalmente, trata-se de arquivos e programas, porém alguns *Shells* mais sofisticados permitem que você faça as operações de programas complexos. *Shells* podem ser gráficos ou em textos, embora a maioria das pessoas hoje em dia ainda pensa em *Shells* como "interface de texto" [Suso 2009].

4.2.1 Expert Sinta

O Expert Sinta é uma ferramenta computacional utilizada para a geração automática de sistemas especialistas, e utiliza técnicas de Inteligência Artificial. Desenvolvido entre 1995 e 1996 pelo Grupo SINTA do Laboratório de Inteligência Artificial da Universidade Federal do Ceará (LIA-UFC), este *software*, livre, ajuda pessoas com pouco conhecimento na área de computação a desenvolver seu próprio sistema especialista e também facilitar o trabalho de implementação de sistemas especialistas [Spirlandelli *et al.* 2011].

O SINTA foi produzido para que o próprio analista de conhecimentos possa programar a base desejada, de modo que o usuário não precise ter quaisquer conhecimentos em programação, somente ideia de como interagir em ambientes visuais [Gilson *apud* Spirandelli *et al.* 2011]. Esta ferramenta possui manual e vários tutoriais na internet com ajuda e explicações de fácil compreensão. A última modificação do projeto realizada no *software* foi em 02/05/1998 e após foi descontinuado [Spirandelli *et al.* 2011].

4.2.2 Clips

O "*C Language Integrated Production System*" (CLIPS) originou-se no Centro Espacial Johnson da NASA em 1984, foi lançado pela primeira vez em 1986, e é submetido a contínuo aperfeiçoamento e melhorias desde então. Ele é uma ferramenta de desenvolvimento de sistemas especialistas que fornece um ambiente completo para a construção de regras de objetos baseados em sistemas especialistas.

O CLIPS fornece uma ferramenta coesiva para gerenciar uma grande variedade de conhecimento com suporte a três diferentes paradigmas de programação: (1) Baseado em Regras: representa o conhecimento como heurísticas, especificando um conjunto de ações a ser executado para uma determinada situação; (2) Orientado a Objetos: permite modelar sistemas complexos como componentes modulares; e (3) Procedural: capacidades semelhantes às encontradas em linguagens como C, Pascal, Ada e LISP [Center 1993; Riley 2008]. É escrito em C e sua portabilidade e velocidade foram mantidas, mesmo instalando em diferentes computadores sem mudança em seu código-fonte. Ele pode ser portado para qualquer máquina que possua um compilador C compatível com o padrão ANSI e seu código-fonte que pode ser modificado de acordo com as necessidades específicas de seus usuários [Center 1993].

É mantido independentemente da NASA, como *software* de domínio público. Sua disponibilidade para uso comercial é livre e ele é totalmente documentado, possuindo um Manual de Referência e o Guia do Usuário [Laerhoven 2013; Riley 2008].

4.2.3 Jess

O *Java Expert System Shell* (JESS) é uma *Shell* de sistema especialista baseado no CLIPS. Ele é um *software* livre para uso acadêmico e com um devido custo para uso comercial. Tal *software* foi escrito por Ernest Friedman-Hill no *Sandia National Laboratories*. Sua primeira versão foi escrita no final de 1995 e evoluiu consideravelmente desde então até a versão atual 8.0, disponibilizada dia 01 de outubro de 2013. Este avanço não se deu apenas com o desenvolvimento, mas também com auxílio de um grande número de usuários, que enviaram códigos, correções e sugestões para o melhoramento do *Shell*. Alguns destes usuários tiveram seus nomes mencionados em agradecimentos no manual do JESS. Para ajudar a utilização do *software* existem alguns documentos que podem ser encontrados no *site* do *software*, tais como manuais e fórum [Herzberg 2013; Jessrules 2013]. O JESS foi originalmente inspirado pela *Shell* de sistema especialista CLIPS, mas cresceu com forte influência do Java. Com ele é possível construir aplicações que têm capacidade de raciocinar usando o conhecimento que você fornece através de regras declarativas. O JESS é muito rápido e, para alguns problemas, é mais rápido até que o CLIPS [Herzberg 2013; Jessrules 2013].

A linguagem utilizada no JESS ainda é compatível com o CLIPS, sendo possível intercambiar *scripts* feitos para CLIPS e JESS. Tal qual o CLIPS, o JESS usa o algoritmo Rete para processar regras, um mecanismo muito eficiente para resolver o difícil problema de muitos-para-muitos acertos. Além disso, foram adicionadas algumas capacidades como encadeamento regressivo e a habilidade para manipular e raciocinar diretamente sobre objetos Java. JESS também é um poderoso ambiente para *scripting* em Java, no qual é possível criar objetos Java e chamar métodos Java sem compilar nenhum código Java [Herzberg 2013; Jessrules 2013].

4.2.4 Netica

O Netica é um programa fácil de usar, poderoso e completo, que serve para trabalhar com redes de Bayes e diagramas de influência. Esta rede de Bayes fornece um método para representar as relações entre proposições ou variáveis, mesmo que os relacionamentos envolvam incerteza, imprevisibilidade ou imprecisão [Norsys 2013].

Este *Shell* foi desenvolvido pela *Norsys Software Corp.* no Canadá e utiliza redes de crença para realizar vários tipos de inferência usando algoritmos modernos e rápidos. Dado um novo caso, pelo qual o usuário tem conhecimento limitado, Netica encontrará os valores ou probabilidades apropriadas para todas as variáveis desconhecidas [Norsys 2013]. O Netica tem uma interface de usuário intuitiva e atualmente novos recursos e capacidades estão em desenvolvimento. Ele possui uma versão grátis, e esta versão demo está completa, mas é limitada no tamanho do modelo [Norsys 2013].

O princípio do Netica está em criar "nós", a partir dos quais são definidas as variáveis e seus atributos (parte qualitativa) e tabelas (parte quantitativa), caracterizadas por valores de probabilidades associadas às variáveis. Este *Shell* é composto de dois ambientes de trabalho, o *Netica Application* e o *Netica API*. O primeiro é a interface gráfica onde a base de conhecimento é visualizada na forma de rede, como os "nós", representando as variáveis e atributos; arcos que representam as dependências causais entre as variáveis e valores de probabilidade que podem ser visualizados através de tabelas. Já o último é uma biblioteca completa de funções escritas na Linguagem C, para criar "nós", adicionar *links*, realizar inferências, compilar e gravar [Toledo *apud* Specht 2003].

4.3 Jogos de Apoio ao Ensino de Inteligência Artificial

O uso de jogos como ferramenta de apoio ao ensino de IA tem duas abordagens: uma focada no desenvolvimento de jogos que auxiliem a ensinar os conceitos e outra focada em aplicar os conceitos de Inteligência Artificial no desenvolvimento de jogos.

Um exemplo desta segunda abordagem vem chamando a atenção dos pesquisadores são os ambientes interativos tridimensionais. Nos últimos anos a Inteligência Artificial tem se tornado parte essencial para estes jogos [Resende 2004]. Na medida em que os jogos se tornam mais complexos, e os consumidores exigem personagens e oponentes controlados por computador, os programadores são obrigados a colocar maior ênfase no desenvolvimento de seus jogos [Vegso 2013].

Neste contexto, os jogos vêm tendo uma grande influência no desenvolvimento da inteligência artificial, pois conforme estes vão se desenvolvendo, os jogadores requerem jogos mais divertidos, com qualidade gráfica melhor e desafios novos, e isso faz com que novas técnicas de inteligência artificial sejam desenvolvidas para a criação dos mesmos.

O fascínio que os jogos exercem sobre as pessoas acaba fazendo com que o mesmo se torne uma forma atraente de ensinar. Porém há uma grande diferença entre jogos educacionais e jogos para diversão, sobretudo pelo fato de que os jogos educacionais da atualidade são bastante chatos, se comparados aos jogos comerciais. A fim de vencer esta oposição, há uma ideia de permitir e facilitar o aprendizado com jogos, ao invés de ensinar com eles. Não sendo obrigado a aprender e sim, estando envolvido com o jogo, tendo, assim, mais probabilidade de aprender [Mattar 2010]. Outro ponto que divide os jogos do aprendizado tradicional é o modo de lidar com o erro. A função do fracasso em videogames é bem diferente do que na escola, onde não são integradas a colaboração e a competição, como ocorre nos jogos. Nos jogos, o custo do fracasso geralmente é diminuído, pois é possível recomeçar do último jogo salvo,

após o jogador fracassar. Ademais, aqui o fracasso é encarado como uma forma de aprendizado, onde na próxima vez arriscará experimentar novas hipóteses, o que seria muito mais complicado de se fazer onde o custo do fracasso é maior, ou que nenhum aprendizado origina-se do fracasso [Mattar 2010].

4.3.1 RoboWar

RoboWar é um projeto de jogo educativo nas áreas de sistemas multiagente, programação paralela e inteligência artificial. Este jogo simula a guerra entre exércitos de robôs que lutam para conquistar Marte, onde cada robô mantém seu próprio estado e é controlado por uma IA individual. Robôs do mesmo exército compartilham os mesmos algoritmos de IA e têm de cooperar para vencer outros pretendentes [Ssd 2014].

Os alunos são convidados a desenvolver os seus algoritmos de IA e apresentá-los para um desafio no campo de batalha. Assim, eles começam a introduzir as ideias de sistemas multiagentes e IA. Tecnicamente, a implementação dos algoritmos é fácil, pois requer apenas uma função em C++ ou *Angel Script*, linguagem da versão atual do simulador. Assim, os alunos do primeiro ano já podem participar deste projeto. No entanto, os aspectos educacionais do projeto são mais amplos. Para os alunos avançados podem ser oferecidos as seguintes tarefas: implementação paralela de um simulador de jogo para sistemas SMP ou *clusters*; desenvolvimento e aperfeiçoamento de visualização do *software*; desenvolvimento de modelos melhorados do jogo; integração do simulador distribuído (baseado em Linux, por exemplo) e visualizador (baseado em Windows) [Ssd 2014].

4.3.2 Robocode

O jogo Robocode foi desenvolvido inicialmente por Matthew A. Nelson no final do ano de 2000, como um trabalho pessoal, e em julho de 2001, tornou-se um trabalho profissional, quando foi levado para a IBM, na forma de um *AlphaWorks*. No início de 2005, o Robocode foi levado para *SourceForge* como *Open Source* com a versão 1.0.7, e a partir daí, o desenvolvimento do Robocode ficou um pouco parado. Neste meio tempo, a comunidade Robocode começou a desenvolver suas próprias versões, a fim de se livrar de problemas (*bugs*) e colocar novos recursos no jogo. Estas melhorias eram disponibilizadas no *OpenSourceRobocode/Contributions* e depois para *RobocodeNG* projeto de Flemming Larsen N. [Robowiki 2014].

Robocode, o nome já diz “código robô”, é um jogo de programação, que tem como objetivo ajudar o usuário a aprender a linguagem Java e curtir enquanto programa (joga). O jogo é baseado na programação do seu robô para competir contra outros robôs em uma arena. Após o robô estar programado, o jogador não poderá atuar em tempo real no jogo, por isso se deve fazer uma programação com uma boa inteligência artificial, para o robô saber qual reação deve tomar a partir do evento que está acontecendo na batalha [Robowiki 2014].

4.3.3 Robocup

A ideia de robôs jogando futebol foi mencionada pela primeira vez pelo Professor Alan Mackworth , em um artigo intitulado "*On Seeing Robots*", que significa “Ao ver robôs”, apresentada em 1992 e, mais tarde publicada no livro “Visão do computador”. No

mesmo ano, pesquisadores japoneses organizaram um *workshop* sobre "Grandes Desafios em Inteligência Artificial", onde houve grandes discussões sobre usar o jogo de futebol para promover a ciência e tecnologia. Em 1993, então, um grupo de pesquisadores lançou uma competição de robótica, provisoriamente chamado o robô *J-League*. Dentro de um mês, no entanto, receberam propostas de pesquisadores fora do Japão, solicitando que a iniciativa fosse alargada como um projeto conjunto internacional. Dessa forma, o projeto foi rebatizado como RoboCup. Seus primeiros jogos oficiais e conferência foram realizados em 1997 com grande sucesso. Mais de 40 equipes participaram (real e simulação combinada), e mais de 5.000 espectadores compareceram [Robocup 2014].

O objetivo do RoboCup é ser utilizado como um veículo para promover a robótica e pesquisas IA, oferecendo um desafio atraente publicamente, mas formidável. Ele é projetado para atender a necessidade de lidar com as complexidades do mundo real, embora em um mundo limitado, mantendo um tamanho de problema e custo da pesquisa acessíveis. O mesmo oferece uma tarefa de pesquisa integrada que cobre grandes áreas de IA e robótica [Robocup 2014].

5 Criação do Portal

Nesta fase, um portal foi desenvolvido utilizando o *Framework* Joomla. Este portal é meramente informativo, trazendo dados sobre inteligência artificial, falando um pouco sobre a matéria de IA em si (Figura 2), tendo como objetivo chamar atenção dos estudantes de IA sobre a importância da matéria para sua carreira.

Figura 2 - Tela principal do portal atualmente.

O portal mapeia alguns *Shells*, *Frameworks* e jogos que podem auxiliar e facilitar seus estudos, trazendo informações, artigos, trabalhos e conteúdos destas ferramentas, como mostra a Figura 3.

Figura 3 - Área de download.

A princípio, o mesmo segue publicado no endereço http://www.cet.adm.br/hls, mas pretende-se dar continuidade ao projeto do portal, publicando-o em um endereço definitivo e realizando outras aplicações para este manter-se atualizado periodicamente, de maneira automática.

6 Conclusão

Concluiu-se com a pesquisa de campo que a maioria das pessoas que a responderam teve pouco ou nenhum contato com as ferramentas abordadas nesta pesquisa. E a maioria dos que tiveram contato traz um retorno positivo sobre a utilização destas no aprendizado.

Já com a análise dos currículos das universidades notou-se a falta da disciplina de Inteligência Artificial em alguns cursos relacionados à computação. Em contrapartida as universidades que oferecem esta disciplina apresentavam bastante divergência em seus conteúdos abordados.

Em relação ao portal desenvolvido pelo autor, algumas dificuldades para elaboração do mesmo foram encontradas, porém puderam rapidamente ser solucionadas através de pesquisas e consultas a pessoas com maior conhecimento na área. Dentre estes obstáculos pode-se citar a publicação do portal na *web,* já que foi o mais difícil de se entender.

Com todas as pesquisas realizadas e com o desenvolvimento deste trabalho como um todo, foi possível notar que o conteúdo da disciplina de Inteligência Artificial é de extrema importância para estudantes de cursos relacionados à computação, visto que isto os auxilia no planejamento e desenvolvimento de *softwares* e produtos que ajudarão pessoas no futuro.

Referências

Balachandran, Bala M. Developing Intelligent Agent Applications with JADE and JESS. P. 236-244. University of Canberra, Autrália, 2008.

Bellifemine, Fabio; Caire, Giovanni; Greenwood, Dominic. Developing Multi-Agent Systems with JADE. John Wiley & Sons, Ltd, 2007.

Caire, Giovanni; Bellifemine, Fabio; Poggi, Agostino; Rimassa, Giovanni. JADE: A White Paper. Volume 3, n. 3, 2003.

Center, Lyndon B. Johnson Space. CLIPS Reference Manual: Basic Programming Guide. Version 6.0. Volume 1. 388 p. 1993.

Coppin, B. Inteligência Artificial. 1ª. Ed. Rio de Janeiro: LTC, 2004.

Fayad, M. E. *et al*. Building application frameworks: object-oriented foundations of framework design. New York: J. Wiley, 1999b. 664 p. ISBN 0471248754.

Fayad, M. E. *et al*. Implementing application frameworks: object-oriented frameworks at work. New York: J. Wiley, 1999a. 729 p. ISBN 0471252018.

Fayad, M. E., Johnson, R. E. Domain-specific application frameworks: frameworks experience by industry. New York: J. Wiley, 681 p. ISBN 0471332801, 2000.

Fernandes, Anita Maria da Rocha; KRAUS, Helton Machado. Ferramenta para Ensino da Técnica de Raciocínio Baseado em Casos. 10 p. Simpósio de Excelência em Gestão e Tecnologia, 2012.

Frank, Eibe; Holmes, Geoffrey; Pfahringer, Bernhard; Reutemann; Peter; Witten, Ian H. The WEKA Data Mining Software: An Update. P. 10-18. SIGKDD Explorations, volume 11.

Gaia. jCOLIBRI. Disponível em: <http://gaia.fdi.ucm.es/research/colibri/jcolibri>. Acessado em: 21 out. 2013.

Gamma, E., HELM, R., JOHNSON, R., VLISSIDES, J. Design patterns: elements of reusable object-oriented software. Addison-Wesley, Reading, MA,1995.

Herzberg. Jess. Disponível em: <http://herzberg.ca.sandia.gov/jess//>. Acessado em: 23 out. 2013.

Jade. Jade. Disponível em: <http://jade.tilab.com/>. Acessado em: 22 out. 2013.

Jessrules. Jess. Disponível em: <http://www.jessrules.com/jess/docs/45//>. Acessado em: 23 out. 2013.

Laerhoven, Kristof Van. CLIPS vs Jess. Disponível em: <http://www.comp.lancs.ac.uk/~kristof/research/notes/clipsvsjess/>. Acessado em: 21 out. 2013.

Maia, W.A. Percepção & Inteligência Artificial - Conceitos, Considerações e Arquitetura. Ed. Biblioteca 24 horas, 2012.

Marrone, Paolo. Java Object Oriented Neural Engine: The Complete Guide. 130 p. 2004.

Mattar, J. Games em educação: como nativos digitais aprendem. Pearson, São Paulo, 2010.

Mattsson, M. Object-oriented Frameworks - A survey of methodological issues.

Licentiate Thesis, Department of Computer Science, Lund University, CODEN: LUTEDX/(TECS-3066)/1-130/(1996), also as Technical Report, LU- CS-TR: 96-167, Department of Computer Science, Lund University, 1996.

Mattsson. M. Evolution and Composition Object-Oriented Frameworks, PhD Thesis, University of Karlskrona/Ronneby, Department of Software Engineering and Computer Science, 2000.

Norsys. Netica. Disponível em: <http://www.norsys.com/>. Acessado em: 22 out. 2013.

Perelman, D. An Academic Asks: Is Computer Science Dead? Disponível em: <http://www.eweek.com/article2/0.1895.2104047.00.asp>. Acessado em: 12 abr. 2013.

Pree, W. Design Patterns for Object-Oriented Software Development. Addison-Wesley, 1995.

Ramalho, G. Papel da Inteligência Artificial na formação universitária em computação no Brasil. Disponível em <http://www.di.ufpe.br/~glr/AI/ia-br.doc>. Acessado em: 23 out. 2012.

Resende, S. O. Sistemas Inteligentes - Fundamentos e Aplicações. 1º.ed. Barueri: Manole, 2004. 525 p.

Riley, Gary. CLIPS Rules. 2008.

Robocup. Robocup. Disponível em: <http://www.robocup.org/>. Acessado em: 15 maio 2014.

Robowiki. Robocode. Disponível em: <http://robowiki.net/wiki/Robocode>. Acessado em: 12 maio 2014.

Sbc – Sociedade Brasileira de Computação. Disponível em: <www.sbc.org.br>. Acessado em: 15 abr. 2013.

Specht, V. R. Utilização de Sistemas Especialistas como Ferramenta para Auxiliar o Processo de Ensino-Aprendizagem Focalizando a Interdisciplinaridade. 6p. Petrópolis, 2003.

Spirlandelli, L. P.; Santos, G. H. D.; Rodrigues, L.; Bandos, M. F. C.; Sistemas Especialistas: Um Estudo de caso com o Expert Sinta. 16 p. Revista Eletrônica de Sistemas de Informação e Gestão Tecnológica. Vol. 1. 2011.

Ssd. RoboWar. Disponível em: <http://ssd.sscc.ru/robowar/index.php/en/>. Acessado em: 12 maio 2014.

Suso. Shell. Disponível em: <http://support.suso.com/supki/What_Is_a_Shell%3F_What_Can_I_Do_In_It%3F>. Acessado em: 8 jun. 2014.

Szyperski, C. Component Software: Beyond Programming, Addison-Wesley, ISBN 0-201-17888-5, 1997.

Tanimoto, S.L. The Elements of Artificial Intelligence. 1ª. Ed. Washington: Computer Science Press, 1987.

Vegso, J. Interest in CS as a Major Drops Among Incoming Freshmen. Disponível em http://www.cra.org/CRN/articles/may05/vegso.> Acessado em: 10 abr. 2013.

Waikato, University. Weka. Disponível em: <http://www.cs.waikato.ac.nz/~ml/weka/index.html>. Acessado em: 22 out. 2013.

Witten, Ian H.; FRANK, Eibe. Data Mining: Pratical Machine Learning Tools and Techniques. 2ª ed. Elsevier, 2005.

Zhong, Xiu-yu. The research and application of web log mining based on the platform weka. P. 4073 – 4078. Procedia Engineering 15, 2011.