

# COMPARATIVO ENTRE APIs JAVA PARA SIMULAÇÃO DE ROBÔS

**Raul SILVA BARROS; Bruno Alberth SILVA BARROS; Omar A. C. CORTES**

(1) Instituto Federal de Educação, Ciência e Tecnologia do Maranhão (IFMA)  
Departamento Acadêmico de Informática  
Av. Getulio Vargas, 04 – Monte Castelo – São Luis –MA  
[raul.sdg@hotmail.com](mailto:raul.sdg@hotmail.com), [b.alberthsb@gmail.com](mailto:b.alberthsb@gmail.com), [omar@ifma.edu.br](mailto:omar@ifma.edu.br)

## RESUMO

Atualmente é possível presenciar um número cada vez maior de robôs exercendo inúmeras funções. Porém, construir um robô requer uma grande alocação de tempo e recursos, podendo ocorrer que após o término da construção da máquina perceber-se que não se está de acordo com o que se desejava. Uma forma de visualizar como seria o robô antes mesmo de ser construído, poupando recursos humanos e financeiros, é a utilização de simuladores. Neste artigo são comparados dois simuladores de robôs livres, o Simbad 3D e o Rossum's Playhouse. O comparativo é feito com base na criação de cenários, sensores e programação. No comparativo também se levam em conta a fidelidade física, fidelidade funcional e facilidade de desenvolvimento. Quatro cenários de diferentes dificuldades foram desenvolvidos para a análise exploratória considerada. Ressalta-se que os simuladores considerados foram desenvolvidos em Linguagem Java.

**Palavras-chave:** Comparativo, Java, Simuladores de Robôs, Simbad 3D, Rossum'sPlayHouse

## 1 INTRODUÇÃO

A cada dia que passa as indústrias estão cada vez mais automatizadas, sendo que essa automatização normalmente é feita por robôs. Segundo Russell (2004), robôs são agentes físicos que executam tarefas manipulando o mundo físico. Segundo Thrun (et. al. 2006), a robótica é a ciência de perceber e manipular o mundo físico através de dispositivos controlados por computadores. Sendo assim, robôs podem passar a ocupar posições que antes eram extremamente insalubres para os seres humanos, manipulando um mundo físico considerado perigoso à seres humanos.

Existem basicamente três categorias de robôs: os manipuladores (braços mecânicos), os robôs móveis (veículo móvel não tripulado) e os humanóides, sendo que os humanóides podem ser considerados como uma mistura entre os robôs manipuladores e os móveis. A principal diferença entre os robôs móveis e os manipuladores é que os manipuladores sabem exatamente onde estão, já os robôs móveis não possuem esse conhecimento prévio nem quando e nem como os objetos do seu ambiente se movimentam. Isso os torna particularmente interessantes, pois os mesmos podem executar tarefas em um ambiente dinâmico e altamente não estruturado (Jones, 2004). Um ambiente estruturado é aquele onde a posição dos objetos é sempre conhecida. Já um ambiente dinâmico é aquele onde os elementos que o formam podem mudar de posição.

No contexto da robótica surgem duas necessidades: a de profissionais qualificados que possam desenvolver software para esse tipo de maquinário, e de equipamentos que permitam o desenvolvimento e teste dos robôs. Construir um robô exige um alto custo, devido às peças e softwares necessários para seu funcionamento. Uma alternativa para diminuir esse gasto é a utilização de simuladores, onde tanto o ambiente onde o robô irá atuar quanto seu comportamento podem ser simulados e testados.

Por exemplo, no trabalho de Tokunaga (Tokunaga et al., 2005) utiliza-se um simulador de robô para planejar a movimentação de um braço mecânico de forma rápida com um custo baixo. O trabalho de Melo (et al., 2008) utiliza um simulador para construir um robô móvel. Já o trabalho de Kulkarni (et al., 1991) utiliza um simulador para fazer o teste de sensores e o planejamento dos caminhos percorridos pelo robô. Um ponto que deve ser destacado é que a utilização de simuladores permite uma forma prática e simples de experimentar diferentes aspectos e configurações na construção de robôs. Além disso, a simulação pode ser pausada para que se faça uma inspeção mais minuciosa dos resultados sem afetar o resultado final.

Existem vários simuladores comerciais, como por exemplo, o *Webot* (Webot Project, 2010), o *Microsoft Robotic Studio* (Robotic Studio, 2010), etc. O *Webot* é um simulador profissional, sendo que os robôs podem ser escritos em Java, C++, Python ou Matlab em qualquer plataforma. Enquanto o *Microsoft Robotic Studio* é um simulador comercial da Microsoft para plataforma Windows, possui já uma série de hardwares implementados e permite a criação de robôs em C#, Visual basic e .NET.

Por outro lado, existem os simuladores de código aberto como, por exemplo, o *Gazebo* (Gazebo, 2010), o *LPZRobot* (LpzRobot Project, 2010), o *OpenSim*, etc. *Gazebo* e o *LPZRobot* são simuladores de robôs móveis que podem ser implementados em C++. O *OpenSim* é um simulador para robôs móveis que podem possuir também partes articuladas. Nesses três exemplo, os robôs são codificados em C++. Para linguagem Java foram encontrados somente dois simuladores de código aberto: o *Rossum's Playhouse* (Rossum Project, 2009) e o *Simbad 3D* (Simbad Project, 2010), sendo estes dois últimos os simuladores considerados neste trabalho.

Para cumprir seu objetivo este artigo está dividido da seguinte forma: na Seção 2 são apresetandos os simuladores *Rossum's Playhouse* (RP1) e *Simbad 3D*; na Seção 3 é mostrada o estudo de caso e a metodologia utilizada no comparativo entre os simuladores; na Seção 4 discutem-se os resultados no desenvolvimento dos cenários com relação à fidelidade física, funcional e facilidade de desenvolvimento; e finalmente na Seção 5 apresentam-se as conclusões deste trabalho.

## 2 SIMULADORES DE ROBÔS

### 2.1 Rossum's Playhouse (RP1)

O *Rossum's Playhouse* (RP1) (Rossum Project, 2009) é um simulador de robô de código fonte aberto em 2D, totalmente programado em Java. A principal vantagem é o fato de ter o código aberto, possibilitando a aplicação de novas características pelo próprio usuário. Esta ferramenta é destinada a ajudar os desenvolvedores de aplicação de controle e lógica de navegação, permitindo construir um robô configurável que pode interagir com uma paisagem simulada ou resolver um labirinto virtual. A Figura 1a mostra um cenário desenvolvido no RP1. Como pode ser observado, é um cenário simples de labirinto, onde o robô parte da origem (marcada com H) e se dirige ao destino (marcado com F). Um grande número de desenvolvedores está usando atualmente o simulador para testar e melhorar as estratégias de concorrência para pequenos robôs. Apesar de ser um ambiente simples, o RP1 pode ser configurado para fornecer desenhos muito mais amplos e apoiar-se em qualquer escala realista.



(a) RP1

(b) Simbad 3D

Figura 1 – Cenário dos simuladores

### 2.2 Simbad 3D

O *Simbad* (Simbad Project, 2010) é um simulador de robôs 3D desenvolvido em linguagem Java e com código fonte aberto, com fins científicos e educacionais. É essencialmente dedicado a pesquisadores e programadores que querem uma plataforma simples para estudar Algoritmos de Inteligência Artificial e *Machine Learning*. Este simulador permite aos programadores escreverem seu próprio controlador de robô com os sensores disponíveis e um comportamento em um ambiente modificável. O *Simbad* pode ser modificado sob as condições da GNU General Public Licence. A Figura 1b mostra o ambiente de simulação

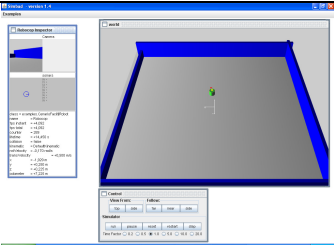
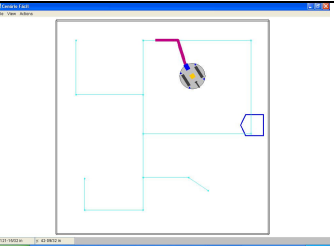
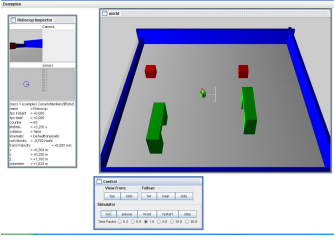
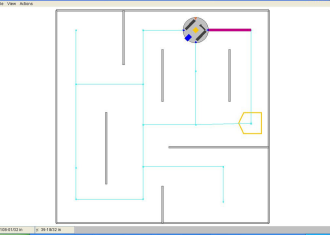
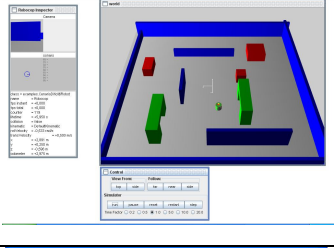
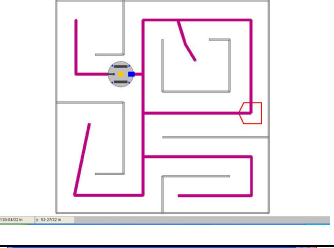
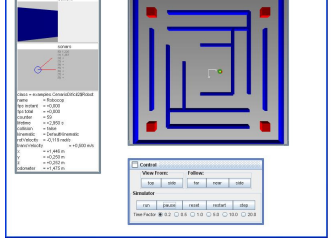
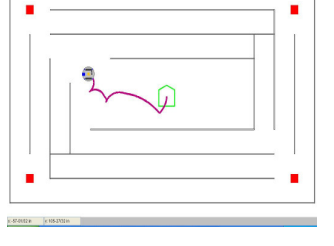
do Simbad 3D, onde se pode visualizar tanto a movimentação do robô quanto a visão da câmera do robô (parte superior esquerda).

### 3 ESTUDO DE CASO

Antes de iniciar o processo de comparação entre os ambientes de simulação, dedicou-se um tempo para o estudo de agentes móveis para obter-se a compreensão sobre como os robôs se comportam, ou seja, como reagem às entradas efetuadas pelo mundo exterior. Dessa forma, três pontos foram estudados: sensores, planejamento de movimento e a movimentação propriamente dita.

Em seguida fez-se o estudo sobre a manipulação das APIs para simulação e de como programar os pontos anteriormente considerados (sensores, planejamento e movimento). O domínio das APIs é um fator essencial para a construção dos mundos virtuais e de alguns tipos de robôs (variando basicamente a quantidade e a posição dos sensores), testando assim o seu comportamento. Além disso, a comparação entre os simuladores tomou como base três dos quatro critérios usados por Craighead (et al., 2007): fidelidade física e funcional, e facilidade de desenvolvimento. O quarto critério, custo, não se aplica, pois as APIs consideradas são livres.

**Tabela 1. Cenários considerados no comparativo**

	<b>Simbad-3D</b>	<b>Rossum's PlayHouse</b>
<b>Dificuldade Baixa</b>		
<b>Dificuldade Mediana</b>		
<b>Dificuldade Alta 1</b>		
<b>Dificuldade Alta 2</b>		

A fidelidade física diz respeito à consonância do simulador com o mundo físico. Por exemplo, avaliar o quão real pode ser o som ou a capacidade de renderização do mundo virtual. Na fidelidade funcional verifica-se a capacidade do simulador de lidar com aspectos físicos como velocidade, aceleração, etc. A facilidade de

desenvolvimento trata de quão fácil é conduzir um exercício de treinamento dentro do simulador. Para comparar os ambientes de simulação foram criados quatro cenários com três níveis de dificuldade: O primeiro possui baixo grau de dificuldade; o outro, grau de dificuldade considerado mediano; e por último dois cenários com nível de dificuldade elevado. Em cada um dos panoramas foi colocado o mesmo agente móvel para se deslocar sem objetivo definido. Os cenários são apresentados na Tabela 1.

Observando-se os cenários nota-se que quanto menor a existência de obstáculos menor seu grau de dificuldade, pois, diminui-se o risco de possíveis colisões do robô com os objetos. Assim sendo tentou-se a criação de cenários o mais semelhante possível entre os simuladores, uma vez que os simuladores apresentam aspectos diferenciados em seus ambientes como no caso do Simbad-3D que é um ambiente tridimensional e apresenta objetos que não podem ser criados no RP1, como por exemplo, os obstáculos do tipo arco em que o agente móvel pode passar por baixo do mesmo. Os testes foram executados em um ambiente Windows.

## 4 RESULTADOS

A criação de cenários no ambiente Simbad é feita via código, sendo seus cenários constituídos basicamente de um plano de fundo sobre o qual são posicionados os objetos/obstáculos (caixas, paredes, arcos) e agentes (robôs). Os cenários são baseados na classe *EnvironmentDescription* e os robôs na classe *Robot*. O posicionamento dos objetos e obstáculos baseia-se em sistemas de coordenadas (x, y, z) pelo fato do ambiente ser tridimensional. O Simbad permite que o usuário utilize funções do seu framework inserindo parâmetros para a configuração/criação do cenário desejado, bastando que o usuário tenha antes uma breve noção do sistema de coordenadas. Um exemplo de código de cenário com muros e caixas é dado a seguir:

```
1. //Código de Exemplo do Simbad 3D
2. import simbad.sim.*; import simbad.gui.*; import javax.vecmath.*;static public
   class MyEnv extends EnvironmentDescription {
3. public MyEnv() {
4.     //Criação de paredes
5.     Wall w1 = new Wall(new Vector3d(9, 0, 0), 18, 1, this);
6.     //Parâmetros do vetor 3D indicam a posição do objeto no espaço (Java 3D)
7.     w1.rotate90(1); add(w1);
8.     Wall w2 = new Wall(new Vector3d(-9, 0, 0), 18, 1, this);
9.     w2.rotate90(1); add(w2);
10.    //Etc...
11.    //Criação de obstáculos - Caixas
12.    Box b1 = new Box(new Vector3d(-7, 0, -7), new Vector3f(1, 1, 1), this);
13.    Box b2 = new Box(new Vector3d(7, 0, -7), new Vector3f(1, 1, 1), this);
14.    add(b1); add(b2);
15.    //Adicionando o robô ao cenário
16.    add(new Robot(new Vector3d(0, 0, 0), "Robocop"));
17.    light1IsOn = true; //Podemos usar essa luz como alvo para o agente (End)
18.    light1SetPosition(6,.7f,5);
19.    ambientLightColor = darkgray;}}
```

Como pode ser observado no código de exemplo acima, a criação de um cenário no Simbad 3D exige apenas conhecimentos de Programação Orientada a Objetos e um pouco de Java 3D. Porém, como os parâmetros são de fácil entendimento, rapidamente o programador pode se adaptar.

O Simbad 3D não permite a criação de vários alvos como o RP1, mas possui, embutido em sua implementação, objetos com funcionalidades similares aos alvos. Os objetos *light1* e *light2* são objetos, transparentes ao programador, que criam pontos de luz que emitem raios infra-vermelhos para detecção dos sensores (*LightSensor* no Simbad). É possível ainda determinar sua posição inicial no cenário. A diferença

básica esta no fato que o Simbad está limitado a apenas 2 alvos nativos (ou seja, já criados e disponíveis pelo Simbad), cabendo ao programador a criação de novos, caso julgue necessário.

O RP1 por ser um ambiente com arquitetura cliente-servidor se divide em cenários (servidores) e clientes (agentes móveis - robôs). A criação de cenários é constituída de obstáculos e agentes, sendo o posicionamento de seus objetos feito mediante o sistema de coordenadas (x, y), já que o ambiente é bidimensional. As informações sobre quais objetos irão compor o cenário, seu tamanho e a coordenada que ele irá pertencer, bem como onde os agentes irão aparecer, quando estes entrarem no cenário, são colocadas em um arquivo texto onde tais informações serão dispostas de forma pré-definida, permitindo que o arquivo seja lido sem problemas pelos métodos das classes Java da API do RP1.

Os objetos são basicamente paredes, obstáculos em formato de polígonos, alvos ou *targets* (pontos de luz ou raios infravermelhos para que o robô possa fazer detecção, podendo servir de objetivo ao agente) e *placements* (a posição inicial do robô), que podem ser usados em conjunto, para dar maior dificuldade aos cenários. O código a seguir mostra um exemplo de cenário usando alguns dos objetos que o RP1 disponibiliza:

```
1 //Exemplo RP1
2 // Criação de paredes
3 wall a { geometry:  -45.0,  0.0,  98.0,  0.0,  0.75; }
4 wall b { geometry:  98.0,  0.0,  98.0,  98.0,  0.75; }
5 // Criação de obstáculos
6 obstacle quadrado1 {
7   polygon: 4, 5.0;
8   offset: 88.0, 90.0;
9   orientation: 135;
10  color: orange;}
11 //Criação de alvos (targets)
12 target F1 {
13   label:      "End";
14   geometry:   -36, 90, 6;
15   color:      red;
16   lineWidth: 3;}
17 //Criação de placements (posição inicial do robô)
18 placement home {
19   label:      "Start";
20   geometry:   89.8, 8.0, 90, 6;
21   color:      green;
22   lineWidth: 3;}
```

A execução do servidor do RP1 é feita mediante a leitura de um arquivo de configurações denominado Rpl.ini que pertence ao diretório *Properties*. Neste arquivo de configuração encontram-se algumas informações muito importantes, como: a porta de conexão do servidor, velocidade de movimentação do agente móvel no cenário e o nome arquivo onde se encontram a codificação do cenário. O código fonte de criação do cenário do servidor encontra-se em arquivo texto localizado no diretório *FloorPlans*, que armazena os arquivos de geração de cenários, uma vez que o usuário pode ter vários e usar o mais adequado para sua necessidade.

Um inconveniente encontrado na construção dos cenários em ambos os simuladores é que sempre que se quiser visualizar como o cenário está deve-se executar o código com as correções nas posições dos objetos. Isso exige certa intuição do programador com relação às coordenadas para que o trabalho fique menos cansativo. Sendo assim, a alteração dos cenários tanto no Simbad quanto no RP1 pode ser bem prática e rápida caso o usuário tenha um breve conhecimento do sistemas de coordenadas e algumas de suas diretrizes

para inclusão ou posicionamento dos possíveis objetos existentes, ou complicada caso contrário. Ambas as ferramentas oferecem suporte a simulação de eventos de colisão, evitando assim que o programador tenha que se preocupar com essa tarefa, o que conseqüentemente leva a um aumento na produtividade.

É possível criar cenários com mais de um agente neles. O Simbad oferece uma gama de agentes não-móveis para que o robô possa interagir como bolas ou pétalas, permitindo que essas sejam empurradas ou coletadas. Além disso, o Simbad permite que dois agentes móveis reconheçam um ao outro, ou seja, simulações com múltiplos robôs. O RP1 também permite que mais de um robô seja usado em simulações, porém não existe interação entre eles.

Com relação a sensores o Simbad-3D apresenta sensores de localização de objetos (sonares) que permitem medir a distância que um objeto está do agente e quais obstáculos estão mais próximos; sensores de colisão (*bumpers*) que fornecem informações de colisão, mas não permitem medidas de distância; e, um sensor que permite mostrar as imagens do cenário nos locais em que o robô se movimenta (câmera sensor).

Além destes sensores o Simbad permite que o usuário crie seus próprios sensores, desde que este possua um conhecimento prévio do esquema de classes Java do ambiente. O RP1 possui sensores de colisão, sensores de distância e sensores que permitem verificar uma localização específica (*target sensor*). Também é possível criar seus próprios sensores no RP1.

De acordo com os critérios de avaliação de Craighead (Craighead et al.,2007), obtiveram-se os seguintes resultados:

- **Fidelidade Física:** consiste no grau de realidade que o ambiente virtual representa o mundo real, como por exemplo: fidelidade áudio-visual do ambiente ao qual se está simulando, podendo variar sons e texturas das imagens em possíveis colisões dos agentes com os objetos do cenário.

Levando em conta este critério o ambiente de simulação 3D, Simbad, é considerado um simulador com grau de fidelidade mediano, apesar de possuir três dimensões, ser composto de uma gama de cores e iluminações, seus cenários são basicamente compostos de figuras geométricas deixando seus cenários não tão reais se comparados ao mundo real, o que impossibilita simulações com robôs reais. O Rossum's PlayHouse no que se refere a este quesito é tido como um simulador com baixo grau de fidelidade física, um fator bem marcante é que o RP1 é um simulador de duas dimensões e o mundo real é composto de formas tridimensionais.

- **Fidelidade Funcional:** é a capacidade que um cenário e os agentes nele contidos possuem de simular ações e reações semelhantes às do mundo real, ou seja, em um ambiente que simule uma pista de corrida, por exemplo, um carro que deseja percorrer este circuito deverá levar em conta fatores como gravidade, atrito entre a pista e o pneu do veículo, o quanto ele irá derrapar ao fazer uma curva fechada entre outros.

Quanto à fidelidade funcional o Simbad e o RP1 são tidos como de baixo grau, pois, não permitem inclusão de fatores que influenciem em seus agentes, como por exemplo, gravidade ou tipos diferentes de solo para influenciar na velocidade dos agentes, permitindo apenas manipular velocidades e mostrar as posições absolutas dos agentes que interagem com os cenários.

- **Facilidade de Desenvolvimento:** é o grau de dificuldade da manipulação e criação de ambientes e agentes nos simuladores, bem como disponibilidade de documentação para ajudar os usuários.

Neste critério o Simbad e RP1 poderiam ser avaliados como de alto grau se não fosse pelo fato que todo o seu material de apoio está disponível apenas no idioma inglês, o que os qualifica com o grau médio neste quesito. Nos sites do Simbad e do Rossum's PlayHouse o usuário encontra materiais de apoio que ensinam os primeiros passos para criação de cenários e agentes, como configurá-los, bem como o funcionamento dos simuladores, seus diagramas de classes, o que permite que o programador após um breve conhecimento dos simuladores criar/modificar com muita facilidade cenários e agentes.

A Tabela 2 apresenta um resumo dos resultados obtidos.

**Tabela 2. Resumo da Análise Comparativa**

	<b>Simbad</b>	<b>Rossum's PlayHouse</b>
<b>Simulação 3D</b>	<b>Sim</b>	<b>Não</b>
<b>Múltiplos Agentes</b>	<b>Sim</b>	<b>Sim</b>
<b>Código Fonte Aberto</b>	<b>Sim</b>	<b>Sim</b>
<b>Custo</b>	<b>Gratuito</b>	<b>Gratuito</b>
<b>Plataformas</b>	<b>Windows/ Linux / Mac</b>	<b>Windows/ Linux / Mac</b>
<b>Robôs Reais</b>	<b>Limitado</b>	<b>Limitado</b>
<b>Fidelidade Física</b>	<b>Mediano</b>	<b>Baixo</b>
<b>Fidelidade Funcional</b>	<b>Baixo</b>	<b>Baixo</b>
<b>Facilidade de Desenvolvimento</b>	<b>Mediano</b>	<b>Mediano</b>

## **5 CONCLUSÃO**

Como já mencionado, implementar robôs reais não é um trabalho trivial e demanda um custo (peças eletrônicas e mecânicas) alto para os desenvolvedores. Este trabalho apresentou um estudo comparativo entre duas APIs Java para a simulação de robôs do código aberto com o objetivo de diminuir os custos demandados.

A análise exploratória através da comparação dos cenários produzidos nos ambientes de simulação de robôs Simbad-3D e Rossum's Playhouse permitiu a observação de especificidades de cada uma das ferramentas. A criação de cenários não é complexa, porém a complexidade do código aumenta na mesma proporção que a complexidade do cenário em ambos os simuladores.

Ambos os ambientes nos permitem a criação de cenários com vários graus de dificuldades para que possamos testar robôs virtuais, bem como apresentam a capacidade de criação de robôs virtuais muito aperfeiçoados, porém o suporte a gráficos de três dimensões do Simbad-3D o deixa um nível a cima do RP1, permitindo um visual mais realista e detalhando, tanto dos cenários quanto de seus robôs.

Em trabalhos futuros, espera-se a construção de simulações mais complexas, com objetivos mais específicos (pegar um objeto, por exemplo) para o reforço nos conceitos de inteligência artificial e na linguagem e o aprendizado de novos simuladores de ambientes de robótica, como o Webots por exemplo.

## **REFERÊNCIAS**

Craighead, J., Murphy, R., Burke, J., & Goldiez, B. (2007). A Survey of Commercial & Open Source Unmanned Vehicle Simulators. IEEE International Conference on Robotics and Automation (pp. 10-14). Roma - Itália: IEEE Press.

Gazebo, Gazebo Home Page, Acesso em 23 de julho de 2010, disponível em:  
<http://playerstage.sourceforge.net/index.php?src=gazebo>

Hugues, L., & Bredeche, N. (2006). Simbad : an Autonomous Robot Simulation Package for Education and Research. The Ninth International Conference on the Simulation of Adaptive Behavior. Roma – Itália.

Jones, J. L., “Robot Programming: A Practical Guide to Behavior-Based Robotics”, New Yor: McGraw Hill, 2004.

Kulkarni, J. A., Byrd, J. S., & Pettus, R. O. (1991). “X-Window Based Graphical of Mobile Robot Simulator for Path Planning and Sensor Fusion Experiments”. 24th Annual Simulation Symposium, (pp. 185 - 192 ).

LpzRobot Project, Research Network for Self-Organization of Robot Behavior, Acesso em 20 de julho de

2010, disponível em: <http://robot.informatik.uni-leipzig.de/software/>

Melo, L. F., Rosário, J. M., Filho, H. F., & Prado, J. G. (2008). “Virtual Simulator for Mobile Robots

Navigation Systems”. 17th IEEE International Conference on Control Applications, (pp. 281-286).

OpenSim Project, A 3D Simulator for Autonomous Robot, Acesso em 20 de julho de 2010, disponível em:

<http://opensimulator.sourceforge.net/>

Robotic Studio, Microsoft Robotic Developer Studio, Acesso em 23 de julho de 2010, disponível em: <http://www.microsoft.com/robotics/>

Rossum Project, Rossum's Playhouse (RP1). (s.d.). Acesso em 06 de julho de 2009, disponível em

Sourceforge: <http://rosum.sourceforge.net/sim.html>

Russel, S. & Norvig, P. (2004). Inteligência Artificial. Rio de janeiro: Elsevier.

Simbad Project, Simba Project Home. Acesso em 20 de julho de 2010, disponível em Sourceforge:

<http://simbad.sourceforge.net>

Thrun, S. & Burgard, W. & Fox, D., “Probabilistic Robotics”, Cambridge: Mit Press, 2006

Tokunaga, H., Matsuki, N., Sawada, H., Okano, T., & Furukawa, Y. (2005). “A Robot Simulator for Manufacturing Tasks on a Component-Based Software Development and Execution Framework”. 6th IEEE International Symposium on Assembly and Task Planning: From Nano to Macro Assembly and Manufacturing, (pp. 162-167).

Webot Project, Webots, Acesso em 20 de julho de 2010, disponível em: <http://www.cyberbotics.com/>

Laue, T., Spiess, K. & Röfer, T. (2005). SimRobot: A General Physical Robot Simulator and Its Application in RoboCup. IX RoboCup 2005: Robot Soccer World Cup (pp. 173–183). Lecture Notes in Artificial Intelligence.

Parker, M. & Parker, G. B. (2007b). The Core: Evolving Autonomous Agent Control. IEEE Symposium in Artificial Life, (pp. 222-228).

TIOBE. (2009). TIOBE Programming Community Index for July 2009. Acesso em 07 de 07 de 2009, disponível em TIOBE Software: <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>