Anais Eletrônicos

Enucomp 2013

enucomp.com.br/2013

IFPI - FMN - UESPI - CEEP

ENUCOMP

ENCONTRO UNIFICADO DE COMPUTAÇÃO EM PARNAÍBA

Organização:
Thiago C. de Sousa
Rodrigo Augusto R. S. Baluz

1a. Edição
Editora FUESPI

REALIZAÇÃO:

FACULDADE MAURÍCIO DE NASSAU
FAZENDO PARTE DA SUA HISTÓRIA

UESPI

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA PIAUÍ Campus Parnaíba

CEEP Informática - Gestão - Saúde - Turismo

Parnaíba - Piauí
2013

enucomp.com.br/**2013**

Organização:
Thiago C. de Sousa
Rodrigo Augusto R. S. Baluz

1a. Edição
Editora FUESPI

REALIZAÇÃO:

Parnaíba - Piauí
2013

E56a

ENCONTRO UNIFICADO DE COMPUTAÇÃO EM PARNAÍBA. (6: 2013: Parnaíba, PI).

Anais do VI ENUCOMP 2013, Parnaíba, PI, 12 a 14 de novembro de 2013: [recurso eletrônico]/ Organização [de] Thiago C. de Sousa e Rodrigo Augusto R. S. Baluz. - Parnaíba: FUESPI, 2013.

159 p.: Il.

ISBN: 978-85-8320-025-3

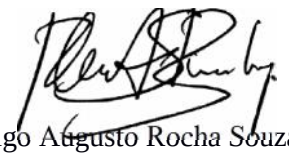
1. Ciência da Computação. 2. Congressos. I. Sousa, Thiago C. de (org.) II. Baluz, Rodrigo Augusto R. S. (org.) III. Título. CDD 001.642

PREFÁCIO

É com muita satisfação que apresentamos os Anais do VI Encontro Unificado de Computação em Parnaíba (ENUCOMP), realizado em Parnaíba – Piauí, entre 12 e 14 de novembro de 2013. Reunimos nesta edição 16 trabalhos científicos abrangendo as mais diversas áreas de conhecimento e pesquisa em Computação. Seis artigos produzidos a partir da submissão de propostas para minicursos no evento. Dez produções adaptadas de Trabalhos de Conclusão de Curso de graduação em Computação, que concorrem às premiações do concurso. Os autores envolvidos nestes trabalhos demonstram a importância da produção científica no seio das instituições, tanto pública como privadas.

O ENUCOMP vem firmando-se como um dos maiores eventos de tecnologia no Estado do Piauí e seus vizinhos. Tem trazido para a comunidade acadêmica e profissional o que há de novo e inovador no campo da Computação. Disponibiliza aos participantes, nesta 6ª edição, uma programação variada e de qualidade. Cinco renomados profissionais e professores pesquisadores apresentaram ao público suas linhas de pesquisa e atuação. Destaque para os profissionais das empresas Oracle Brasil e ThoughtWorksBrazil de São Paulo. Seis minicursos serão ofertados como capacitação aos participantes, em áreas como TV Digital, Cluster de Serviços, Projetos 3D com Blender, Engenharia de Software Orientada a Agentes, Redes Neurais e Robocode. Pela primeira vez o Comitê de Programa Científico traz o Concurso de Trabalhos de Conclusão de Curso, modalidade Graduação. Os 10 trabalhos que concorrem à premiação do concurso versam sobre três grandes áreas da Computação: Inteligência Artificial, Engenharia de Software e Metodologias Ágeis. Ainda, compõem a grade de programação do ENUCOMP 2013 três workshops: Inovação, Ciência e Tecnologia; Jogos Eletrônicos e Robótica.

A organização de um evento do porte e da importância do ENUCOMP só pode ser realizada se contar com a ajuda de uma equipe qualificada e dedicada. Gostaríamos de agradecer aos membros do Comitê de Organização Geral pelo trabalho voluntário de excelente qualidade e pelo apoio incansável durante as várias etapas da organização deste evento. Somos muito gratos também ao apoio da Sociedade Brasileira de Computação (SBC) e ao Núcleo de Pesquisa e Extensão em Computação do Delta do Parnaíba (NUPEC Delta). Em particular aproveitamos a oportunidade para agradecer a todos os autores pela escolha de nosso evento como fórum de publicação de seus trabalhos, aos membros revisores do Comitê de Programa Científico que não mediram esforços para avaliar de forma criteriosa os trabalhos submetidos. Finalmente, o mais expressivo agradecimento a todas aquelas pessoas e instituições públicas e privadas que contribuíram e têm contribuído ao longo do tempo para que se mantenha acesa a chama do conhecimento em Computação ao norte do Piauí.



Rodrigo Augusto Rocha Souza Baluz
Coordenação Geral
ENUCOMP 2013

COMISSÃO ORGANIZADORA

Coordenação Geral:

Francisco das Chagas Rocha, Universidade Estadual do Piauí (UESPI)

Francisco Gerson Amorim de Meneses, Instituto Federal do Piauí (IFPI)

Muryell Penafiel Diniz de Aragão, Centro Estadual de Educação Profissional (CEEP)

Rodrigo Augusto Rocha Souza Baluz, Faculdade Maurício de Nassau (FMN)

Equipe de Apoio:

Antônio S. de Sousa, Instituto Federal do Piauí (IFPI)

Átila R. Lopes, Universidade Estadual do Piauí (UESPI)

Cornélia Janayna Pereira Passarinho, Universidade Estadual do Piauí (UESPI)

Francisco das Chagas Coelho do Nascimento, Centro Estadual de Educação Profissional (CEEP)

José Flávio G. Barros, Faculdade Maurício de Nassau (FMN)

Lianna Mara Castro Duarte, Universidade Estadual do Piauí (UESPI)

Mayllon V. da Silva, Faculdade Maurício de Nassau (FMN)

Nécio de L. Veras, Instituto Federal do Ceará (IFCE)

Régis P. Magalhães, Universidade Federal do Ceará (UFC)

Thiago C. de Sousa, Universidade Estadual do Piauí (UESPI)

Comitê de Programa:

Anderson Passos Aragão, Universidade Estadual do Piauí (UESPI)

Átila Rabelo Lopes, Universidade Estadual do Piauí (UESPI)

Cornélia Janayna Pereira Passarinho, Universidade Estadual do Piauí (UESPI)

Denival Araújo dos Santos, Instituto Federal do Piauí (IFPI)

Eyder Franco Sousa Rios, Universidade Estadual do Piauí (UESPI)

Comitê de Programa (cont.) :

Francisco Marcelino Almeida de Araújo, Instituto Federal do Piauí (IFPI)

José Flávio Gomes Barros, Faculdade Maurício de Nassau (FMN)

Lianna Mara Castro Duarte, Universidade Estadual do Piauí (UESPI)

Regis Pires Magalhães, Universidade Federal do Ceará (UFC)

Rodrigo Augusto Rocha Souza Baluz , Faculdade Maurício de Nassau (FMN)

Sérgio Barros de Sousa, Universidade Estadual do Piauí (UESPI)

SUMÁRIO

1. Desenvolvimento de Aplicações para TV Digital Interativa.....7
2. Cluster de Serviços e Alta Disponibilidade com Software Livre.....22
3. Criação de Projetos 3D com Blender.....33
4. Introdução à Engenharia de Software Orientada a Agentes com JaCaMo.....45
5. Introdução a Redes Neurais Artificiais com a Biblioteca ENCOG em Java 66
6. Utilizando Java para Construir e Destruir Robôs.....80
7. Utilização de Heurísticas Bio-inspiradas em Sistemas de Inteligência Coletiva para Otimização Combinatória em Redes Mesh.....91
8. Uma proposta para Classificação de Rotas em Redes de Sensores sem Fio baseada em Sistemas Fuzzy e Otimização por Colônia de Formigas.....97
9. Uma proposta de Verificação Formal em um Processo de Desenvolvimento Orientado pela UML.....104
10. Avaliação do Comprometimento das Equipes para o Impacto das Metodologias Ágeis na Utilização do Desenvolvimento de Software de uma Organização.....110
11. Análise e Implementação de um Sistema para o Gerenciamento de Estágios Curriculares na Faculdade Piauiense- FAP/Parnaíba.....116
12. Desenvolvimento de um Sistema de Controle de Iluminação Fuzzy.....122
13. Controle e Gerenciamento do Consumo de Energia utilizando Arduino.....128
14. Uma Proposta Para Controle De Densidade em Redes de Sensores Sem Fio utilizando Inteligência Computacional.....138
15. Projeto Controle: Prototype of Telerobotic using Python and Arduino 149

Desenvolvimento de Aplicações para TV Digital Interariva

Gleison Brito Batista[1], Fábio de Jesus Lima Gomes[1]

[1]Instituto Federal do Piauí - IFPI
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil
gleison.brito@aluno.ifpi.edu.br, fabiogomes@ifpi.edu.br

Abstract. *This short course aims to disseminate and address concepts involved on iDTV, present the main aspects of the middleware Ginga and its subsystems (Ginga-NCL and Ginga-J), but also address the development of interactive applications using NCL and Lua languages, which will be concepts presented on these languages, as well as the relationship between them. Some examples of applications will be developed, demonstrating the applicability of the concepts discussed.*

Resumo. *Este minicurso visa disseminar e abordar conceitos envolvidos sobre TVDI, apresentar os principais aspectos do middleware Ginga e de seus subsistemas (Ginga-NCL e Ginga-J), como também, abordar o desenvolvimento de aplicações interativas utilizando as linguagens NCL e Lua, onde serão apresentados conceitos sobre estas linguagens, como também o relacionamento entre elas. Alguns exemplos de aplicações serão desenvolvidos, demonstrando-se a aplicabilidade dos conceitos discutidos.*

1. Introdução

No Sistema Brasileiro de TV Digital adota o middleware Ginga, desenvolvido pela PUC-Rio e UFPR, como camada de aplicações. O Ginga é uma camada de software que dá suporte à execução de aplicações interativas nos conversores digitais, esses últimos instalados nas casas dos telespectadores. As aplicações desenvolvidas para TV Digital podem ser declarativas, procedurais ou híbridas (aplicações declarativas com partes procedurais ou o contrário). Essa característica permite que seja escolhida a alternativa mais viável durante o desenvolvimento das aplicações.

O ambiente procedural do Ginga, também chamado de máquina de execução, é responsável pelo suporte a aplicações desenvolvidas na linguagem de programação Java. O ambiente declarativo, também denominado máquina de apresentação, interpreta aplicações desenvolvidas na linguagem declarativa NCL (Nested Context Language). Neste capítulo abordaremos o desenvolvimento de aplicações no ambiente declarativo.

Linguagens declarativas são mais intuitivas que as procedurais, por esta razão são de mais fácil aprendizado e utilização. Ao desenvolver nesse paradigma, o programador informa as tarefas a serem realizadas, não preocupando-se com os detalhes da execução das tarefas. No entanto linguagens declarativas são definidas com um foco específico.

O middleware Ginga (brasileiro) e outros middlewares que possuem os dois ambientes de programação, permitem que seja feita uma ponte entre os ambientes para que se desfrute das facilidades inerentes a cada um.

2. Middleware Ginga

O Sistema Brasileiro de TV Digital (SBTVD) foi implantado no final de 2007 e possui o Ginga como middleware. O Ginga é constituído por dois subsistemas responsáveis pela execução de aplicações interativas: Ginga-NCL e Ginga-J. O módulo Ginga-NCL utiliza uma linguagem declarativa de marcação baseada em XML denominada NCL (Nested Context Language). NCL define a forma como o conteúdo é exibido, a definição dos relacionamentos entre as mídias e permite exibição em múltiplos dispositivos. O módulo Ginga-J utiliza a linguagem de programação Java para o desenvolvimento de aplicações.

No mercado encontramos disponíveis diversos modelos de hardwares responsáveis pela decodificação do sinal digital (set-top boxes). As aplicações desenvolvidas são distribuídas indistintamente para todos eles. Para evitar incompatibilidades entre os modelos existentes e garantir que as aplicações executem bem em todos set-top boxes, desenvolveu-se uma camada de software denominada middleware.

A camada middleware localiza-se ente o sistema operacional e as aplicações. A função do middleware é prover um conjunto de funções com a finalidade de padronizar o desenvolvimento de programas interativos. Com todas aplicações executando as mesmas funções comuns, o middleware fica responsável pela comunicação com os vários hardwares existentes, traduzindo as aplicações desenvolvidas em um padrão comum, para hardwares e sistemas operacionais diferentes.

Como mencionado anteriormente, o middleware adotado no SBTVD é o Ginga, desenvolvido pela PUC-RJ e UFPB. Na Figura 1.1 podemos perceber as partes que compõem o Ginga: Ginga-NCL, Ginga-J e Ginga Common Core.

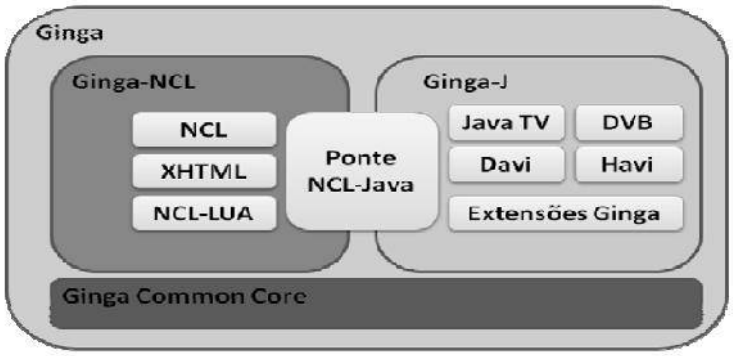


Figura 1.1. Arquitetura do middleware Ginga

Na Figura 1 podemos perceber que aplicações implementadas no ambiente Ginga-NCL podem utilizar as linguagens NCL, HTML e scripts Lua. As aplicações desenvolvidas no ambiente Ginga-J utilizam a linguagem de programação Java. Há também a possibilidade de criação de programas híbridos, implementados utilizando

NCL e Java. Nesse tipo de aplicação o código Java é tratado como uma mídia pelo NCL, além de o Java poder manipular o NCL.

3. Aplicações para TVDI

Aplicações para TV Digital podem estar ou não associadas semanticamente ao conteúdo apresentado no vídeo principal da TV. Também há a possibilidade de poderem definir o sincronismo entre os objetos de mídia que a compõem, inclusive o conteúdo principal, sendo ele vídeo ou áudio.

As aplicações desenvolvidas para TVDI podem estar sempre disponíveis, ou disponíveis durante um período de tempo, como durante a apresentação de um programa com o qual ela se relaciona. Há também aplicações capazes de manter uma relação de sincronismo com outras aplicações, não só apenas com seus objetos de mídia. Esse tipo de aplicação é denominado programa não linear.

A TV analógica apresenta um único caminho sequencial de exibição. De forma contrária, programas não lineares são compostos de múltiplas cadeias de exibição. Durante um determinado momento da exibição de um programa não linear o telespectador pode escolher formas alternativas para sua continuação. Assim um programa deixa de ser representado por uma linha do tempo e passa a ser representado por um grafo.

Em boa parte dos casos, linguagens declarativas são preferenciais no desenvolvimento de programas não lineares. Porém, assim como em programas lineares, o sincronismo entre as mídias, mesmo sem a intervenção do usuário, dever ser tão ou mais importante como a interatividade. O sincronismo entre as mídias deve ser o foco das linguagens declarativas, como o NCL.

O suporte a múltiplos dispositivos de exibição é também uma caraterística importante no suporte à interatividade de um sistema de TV digital. Por meio de múltiplos dispositivos de exibição pode-se transferir a aplicação para algum dispositivo particular do usuário, sem que apareça na TV, não incomodando os demais telespectadores.

4. Linguagem Declarativa NCL (Nested Context Language)

A linguagem NCL (Nested Context Language) é uma linguagem declarativa baseada no modelo conceitual NCM (Nested Context Model) desenvolvida para autoria de documentos hipermídia. Sua estrutura é modular e sua especificação segue os princípios definidos pela W3C.

Base do Ginga-NCL, a NCL define uma separação bem demarcada entre o conteúdo e a estrutura de um documento, provendo um controle não invasivo da ligação entre o conteúdo e seu layout. A linguagem NCL utiliza os conceitos usuais de nós e elos, especificados no modelo NCM, para descrever documentos hipermídia. Os nós (*nodes*) representam fragmentos de informações, que podem ser de mídia ou de contexto, e elos (*links*) são usados para definição de relacionamentos entre os nós [Soares e Rodrigues 2005].

Os nós e elos de documentos hipermídia NCL podem ser aninhados, como ilustrado na Figura 1.2, permitindo uma estrutura mais enxuta e organizada.

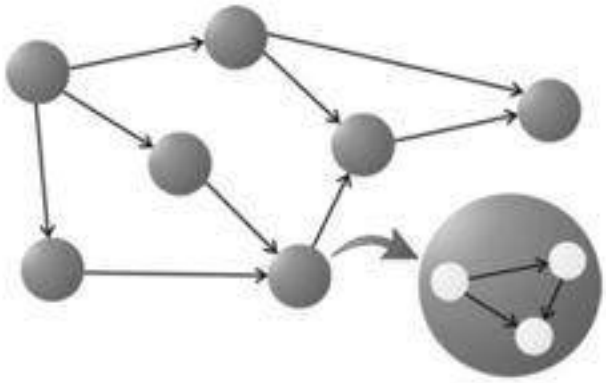


Figura 1.2. Representação do modelo NCM

A estrutura de um documento NCL deve possuir o cabeçalho de definição para arquivos XML, o cabeçalho do programa e o corpo do documento. Na definição de um documento NCL deve-se definir a mídia a ser exibida, onde, como e quando será exibida. A listagem 1.1 mostra a estrutura básica de um documento NCL.

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<ncl id="main" xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile">

   <head>
       <!-- Onde? Aqui são definidas as regiões -->
       <!-- Como? Aqui são definidos os descritores -->
   </head>


   <body>
       <!-- O que? Aqui são definidos os nós de mídia -->
       <!-- Quando? Aqui são definidos os elos -->
   </body>

</ncl>
```

Listagem 1.1. Estrutura básica de um documento NCL

A estrutura básica de um documento NCL é definida por um documento raiz, denominado <ncl>, e seus elementos filhos: <head> e <body>, respectivamente o cabeçalho e o corpo do documento.

4.1. Definindo Regiões

Regiões são áreas da tela onde as mídias da aplicação são exibidas. A tela pode ser uma TV, celular ou outro equipamento compatível com a tecnologia. As regiões são definidas no cabeçalho do programa (<head>) dentro da base de regiões (<regionBase>). Na listagem 1.2 é apresentado um exemplo de como são definidas regiões em documentos NCL.

```
<head>
   <regionBase>
      <region id="rgTela" height="100%" width="100%">
         <region id="rgVideo01" height="50%" width="50%" left="10%"
```

```
top="30%"/>
      </region>
   </regionBase>

</head>
```

Listagem 1.2. Definição de regiões

Na listagem 1.2 ao definir uma região é atribuído um identificador único (atributo id). O id será referenciado nos descritores das mídias associadas às regiões. Uma região pode ser aninhada a outra região, para facilitar o posicionamento e o dimensionamento relativo entre elas. A linguagem NCL define os seguintes atributos do elemento <region>:

- **id**: identificador da região. Deve possuir um valor único e será utilizado quando for feita alguma referência à região;
- **left**: define a coordenada horizontal à esquerda da região;
- **right**: define a coordenada horizontal à direita da região;
- **top**: define a coordenada vertical superior da região;
- **bottom**: define a coordenada vertical inferior da região;
- **width e height**: dimensões horizontal e vertical da região;
- **zIndex**: é utilizado para indicar a sobreposição de camadas.

4.2. Definindo Descritores

O elemento descritor (<descriptor>) é responsável por associar uma mídia a uma região. Por meio dos descritores são determinadas as propriedades iniciais para a exibição de um objeto de mídia, como por exemplo, o volume de um áudio, a transparência de uma imagem. Os descritores são definidos no cabeçalho do programa (<head>), dentro da base de descritores (<descriptorBase>). Na listagem 1.3 é apresentado um exemplo de definição de descritores.

```
<descriptorBase>
   <descriptor id="dsTela" region="rgTela"/>
   <descriptor id="dsVideo01" region="rgVideo01"/>
</descriptorBase>
```

Listagem 1.3. Definição de descritores

O elemento <descriptor> possui um identificador único identificado pelo atributo id. O atributo id é utilizado quando se faz referências ao descritor. O atributo region define a região onde a mídia será apresentada. Também podemos definir a duração da exibição da mídia por meio do atributo explicitDur .

A linguagem NCL também define o elemento opcional <descriptorParam>,onde podemos iniciar uma propriedade com um valor. Na listagem 1.4 é demonstrada a utilização desse elemento na inicialização da propriedade transparency (transparência).

```
<descriptor id="dsVideo01" region="rgVideo01">
   <descriptorParam name="transparency" value="90%"/>

</descriptor>
```

Listagem 1.4. Definição de um parâmetro de descritor

Dentre os parâmetros reservados para mídias visuais podemos destacar:

- **fit:** modifica o elemento visual para que se apresente melhor na região. Pode receber os valores *fill*, *hidden*, *meet*, *meetBest* ou *slice*.
- **bounds:** modifica a posição e as dimensões do elemento de mídia. É definida por quatro parâmetros: *left*, *top*, *width* e *height*, com valores em percentagem ou pixels.
- **transparecy:** assume um valor real entre 0 e 1, onde 0 significa totalmente opaco e 1 totalmente transparente.

4.3. Definindo nós de mídia

Em NCL um nó de mídia representa o conteúdo que desejamos apresentar, como vídeos, imagens, textos, dentre outros. Todo nó de mídia é definido em um contexto, onde o elemento <body> é o contexto que contém todos os nós do documento (mídias ou outros contextos). A Figura 1.3 representa seis nós de mídia aninhados, onde cinco estão dentro de contextos aninhados ao elemento <body>.

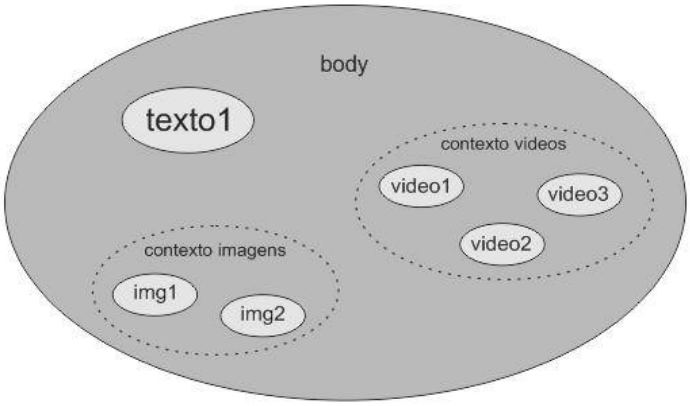


Figura 1.3. Representação de nós de mídia e de contextos

Objetos de mídia são representados pelo elemento <media>, que deve ser inserido dentro do elemento <body>. O elemento <media> contém os atributos id (identificador único), src (diretório onde se encontra o conteúdo do objeto), type (o tipo do objeto referenciado) e descriptor (identificador do descritor que controlará a exibição da mídia). Na listagem 1.5 é apresentado um exemplo de definição de um nó de mídia.

```
<body>
   <media id="video01" descriptor="dsVideo01"
   type="video/mpeg" src="media/video01.mpg"/>

</body>
```

Listagem 1.5. Representação de nós de mídia e de contextos

Na tabela 1.1 estão listados os principais tipos de mídias e suas respectivas notações para o atributo type.

Tabela 1.1. Principais tipos de mídia suportados pela linguagem NCL.

Valor de type	Extensão da mídia

text/html	.htm, .html
text/css	.css
text/xml	.xml
image/bmp	.bmp
image/png	.png
image/gif	.gif
image/jpeg	.jpg
audio/basic	.wav
audio/mp3	.mp3
audio/mp2	.mp2
audio/mpeg4	.mp4, .mpg4
video/mpeg	.mpeg, .mpg
application/x-ginga-NCLua	.lua

Como vimos anteriormente, contextos são utilizados como uma forma de organizar e estruturar um documento hipermídia NCL. Contextos são representados pelo elemento <context>. Na listagem 1.6 é demonstrada a utilização desse elemento.

```
<context id="contexto01">
<port id="start01" component="video01" />

   <media id="video01" descriptor="dsVideo01"
   type="video/mpeg" src="media/video01.mpg"/>

   <media id="video02" descriptor="dsVideo02"
   type="video/mpeg" src="media/video02.mpg"/>

</context>
```

Listagem 1.6. Definição de um elemento contexto

4.4. Âncoras de conteúdo e propriedades

Um nó de mídia ou de composição também pode ter âncoras associadas. NCL define dois tipos de âncoras: âncoras de conteúdo e âncoras de propriedades. Âncoras de conteúdo definem intervalos de tempo da mídia, enquanto âncoras de propriedades definem propriedades relacionadas aos objetos de mídia.

Âncoras de conteúdo são comumente utilizadas para sincronizar mídias. São definidas pelo elemento <area> dentro do elemento <media>. O elemento <area> deve possuir um identificador único (atributo id). Na listagem 1.7 é demonstrado um exemplo da utilização do elemento <area>.

```
<media id="video02" descriptor="dsVideo02" type="video/mpeg"
src="media/video02.mpg">
   <area id="aTrecho01" begin="10s" end="20s"/>
   <area id="aTrecho02" begin="20s" end="30s"/>
   <area id="aTrecho03" begin="30s" end="35s"/>

</media>
```

Listagem 1.7. Exemplo de âncoras de conteúdo

Âncoras de propriedades definem propriedades das mídias e permitem que sejam alterados seus valores. São definidas pelo elemento <property> possuem apenas dois atributos: nome (*name*) e valor (*value*). Na listagem 1.8 exemplifica a definição das propriedades relativas ao dimensionamento de um nó de mídia.

```
<media id="video01" descriptor="dsVideo01"  type="video/mpeg"
src="media/video01.mpg">

   <property name="top"/>
   <property name="left"/>
   <property name="height"/>
   <property name="width"/>

</media>
```

Listagem 1.8. Exemplo de âncoras de propriedade

4.5. Sincronizando com elos e conectores

Após definir os elementos de mídia e suas respectivas regiões e descritores deve-se definir o fluxo da aplicação, quando os elementos serão apresentados. Primeiro define-se o primeiro nó de mídia a ser apresentado. Para isso utiliza-se o conceito de porta, representado pelo elemento <port> do NCL. O elemento <port> é o ponto de interface de um contexto, isto é, por meio deste elemento temos acesso aos nós internos.

Na definição de uma porta temos dois atributos: *id* (identificador único) e *component* (nó de mídia a ser acessado na chamada ao contexto). Na listagem 1.9 é apresentado um exemplo de definição de uma porta.

```
<port id="start" component="contexto01"/>
```

Listagem 1.9. Exemplo de definição de porta

Em aplicações hipermídia comumente devem ser especificados os relacionamentos entre as mídias. Em NCL o elemento conector é responsável por definir esse relacionamento. Relacionamentos são definidos por mecanismos de casualidade definidos no elemento conector. Em uma relação causal, uma condição deve ser satisfeita para que ações possam ser disparadas. Conceitualmente um conector pode representar qualquer tipo de relação hipermídia, como relações de referência e sincronização.

Os conectores são definidos em uma base de conectores <connectorBase>, localizada no cabeçalho da aplicação (<head>). Um elemento <causalConnector> representa uma relação causal, onde uma condição deve ser satisfeita para que ações possam ser disparadas. No conector são definidos os papeis (*role*) que os nós de origem e destino assumem, e as ações (*action*) que serão realizadas quando o conector for utilizado. Na listagem 1.10 encontra-se um exemplo de definição de um conector.

```
<causalConnector id="onEndStart">
   <simpleCondition role="onEnd" />
   <simpleAction role="start" />
</causalConnector>
```

Listagem 1.10. Definição de conector

Usualmente os conectores são criados em um arquivo separado para serem aproveitados em outras aplicações. Para importar um conector deve ser especificado um alias para referência na criação de elos. Também deve ser especificado o atributo *baseURI*, onde deve ser informado o endereço do arquivo. A listagem 1.11 demonstra uma importação de base de conectores.

```
<connectorBase>
   <importBase documentURI="ConnectorBase.ncl" alias="conn"/>
</connectorBase>
```

Listagem 1.11. Importando base de conectores

Um elo no modelo NCM que possui duas propriedades adicionais: um conector e um conjunto de associações a esse conector [Soares e Rodrigues 2007]. Os elos são definidos na linguagem NCL pelo elemento <link>, onde os objetos de mídia são associados aos conectores. Um elo é criado aplicando aos nós de mídia ou contexto os papeis estabelecidos pelo conector. A listagem 1.12 descreve a criação de um elemento <link>.

```
<link xconnector="conn#onEndStart">
   <bind role="onEnd" component="video01"/>
   <bind role="start" component="video02"/>
</link>
```

Listagem 1.12. Definição de conector

5. Linguagem procedural Lua

A linguagem de programação Lua foi desenvolvida em 1993 na PUC-Rio pelos professores Roberto Ierusalimschy, Luiz Henrique de Figueredo e Waldemar Celes. Lua foi projetada com o propósito de ser utilizada em conjunto com outras linguagens, não sendo comum encontrar programas puramente escritos em Lua.

Lua é uma linguagem de fácil aprendizado, que contém poucos comandos primitivos. Lua também apresenta um alto grau de portabilidade, podendo ser executada com todas as suas funcionalidades em diversas plataformas, como aplicações para TV Digital, aplicações web, aplicações desktop ou jogos.

5.1. Convenções léxicas

Identificadores na linguagem Lua podem ser representados por qualquer cadeia de caracteres, dígitos e sublinhados, porém não podem iniciar com dígitos. Os identificadores são utilizados para nomear funções, variáveis ou campos de tabelas. Lua possui algumas palavras reservadas, que não podem ser utilizadas em identificadores.

Tabela 1.2. Palavras reservadas.

And	Return	True
False	Do	elseif
In	Nil	If
Repeat	Then	Or
Break	Else	until

False	Function	while
Local	Not	

A linguagem também define operadores aritméticos, relacionais, de concatenação e de tamanho.

Tabela 1.3. Operadores.

+	-	*	/	%	#	^
==	~=	<=	>=	<	>	=
(	)	{	}	[	]	;
:	,	.	..	...		

Lua é uma linguagem *case sensitive*, ou seja, o identificador ENUCOMP é interpretado de uma forma diferente do identificador enucomp.

Comentários são definidos como um hífen duplo (--) (comentários de uma linha) ou por um texto escrito entre duplos colchetes, precedidos do hífen duplo (--[[]]). Na listagem 1.13 é demonstrada a criação de comentários.

```
-- Comentário de uma linha
--[[Exemplo de comentário
em múltiplas linhas]]
```

Listagem 1.13. Exemplos de comentários em Lua

5.2. Variáveis

Variáveis são definidas como espaços de memória destinados a armazenar valores. Por padrão na linguagem Lua as variáveis são globais. Para declarar-se uma variável local, deve-se utilizar a palavra *local* precedendo o identificador da variável.

Para atribuir-se valores às variáveis utiliza-se o operador de atribuição =. Lua tem suporte a atribuições múltiplas, onde os valores excedentes são descartados e se houver mais variáveis que valores, as variáveis excedentes recebem um valor nulo (*nil*). Na listagem 1.14 encontram-se alguns exemplos de atribuições.

```
nome = "ENUCOMP"
local name = "enucomp"
x = 3 -- x recebe o valor 3
x, y = 3, 1 -- x recebe o valor 2 e y recebe o valor 1
x, y, z = 3, 2 -- x recebe o valor 3, y recebe o valor 2 e z recebe nil
x, y = 3, 2, 1 -- o valor 1 é descartado
```

Listagem 1.14. Exemplos de atribuições

5.3. Tipos de valores

Lua possui tipagem dinâmica, ou seja, os valores possuem os tipos, não as variáveis. Nessa linguagem encontram-se apenas oito tipos de valores: *nil*, *boolean*, *number*, *string*, *function*, *userdata*, *thread* e *table*. Para verificar o tipo de um valor é utilizado o comando *type*.

5.3.1 Nil

Representa a ausência de valor. Variáveis não inicializadas possuem o valor *nil*.

```
local nomeVariavel
print(type(nomeVariavel)) –nil
```

Listagem 1.15. Variável não inicializada

5.3.2 Number

Representa números inteiros e reais. Equivale a um double de dupla precisão.

```
local valor = 1
print(type(valor)) –number
```

Listagem 1.15. Variável não inicializada

5.3.3 String

Strings representam cadeias de caracteres. Podem ser definidas pelo uso de aspas simples (‘’), aspas duplas (“”) ou no formato longo ([[]]) para cadeias que se extendem por várias linhas. Também podem ser inseridos caracteres de escape como na linguagem C.

```
name = 'enucomp'
name = "enucomp"
name = [[enucomp]]
```

Listagem 1.16. Definição de strings

5.3.4 Function

Representam as funções. Funções podem ter nenhum ou vários parâmetros e retornar um ou vários valores.

```
foo = function(x)
        print "exemplo funcao"
end
```

Listagem 1.17. Exemplos de definição de funções

5.3.5 Table

Por meio do tipo *table* pode-se representar vetores, registros, grafos, árvores, etc. Tabelas podem conter vários tipos de valores, exceto *nil*. Na listagem 1.18 encontramos um exemplo de definição de tabelas em Lua.

```
tabela = {} --tabela vazia
tabela = {1, "enucomp", true} --tabela com três valores diferentes
tabela = {name = "enucomp"} --tabela com índice personalizado
```

Listagem 1.18. Definição de tabelas

5.4 Operadores

A linguagem Lua fornece um conjunto de operadores aritméticos (tabela 1.4), relacionais (tabela 1.5), lógicos (tabela 1.6) e de concatenação (tabela 1.7).

Tabela 1.4. Operadores aritméticos.

Operadores	Significado	Exemplo
+	Adição	a + b
-	Subtração	a – b
*	Multiplicação	a * b
/	Divisão	a / b
%	Módulo	a % b
^	Exponenciação	a ^ b
~	Negação	~ a

Tabela 1.5. Operadores relacionais.

Operadores	Significado	Exemplo
==	Igualdade	a == b
~=	Diferença	a ~= b
<	Menor que	a < b
>	Maior que	a > b
<=	Menor ou igual	a <= b
>=	Maior ou igual	a >= b

Tabela 1.6. Operadores lógicos.

Operadores	Significado	Exemplo
And	E	a and b
Or	OU	a or b
Not	NÃO	not a

Tabela 1.7. Operadores de concatenação.

Operadores	Significado	Exemplo
..	Concatenação	a..b

5.5 Estruturas de Controle

Estruturas de controle são utilizadas quando é necessário executar algumas instruções baseadas em condições. Lua contém algumas instruções para controle do fluxo da aplicação.

5.5.1 If..then...else

O comando *if* representa uma tomada de decisão de dois ou vários caminhos na execução. Se a expressão analisada for verdadeira o bloco *then* é executado, senão o bloco *else* é executado.

5.5.2 While

O *while* realiza um teste lógico na entrada da estrutura. Enquanto o teste for verdadeiro o bloco executa.

5.5.3 Repeat

A estrutura *repeat-until* realiza um teste lógico em *until*. O bloco termina quando a condição do *until* for satisfeita.

5.5.4 For

O comando *for* numérico é formado por três parâmetros, onde o primeiro é o valor de inicialização da contagem, o segundo é o valor a ser atingido para parar a execução do loop e o terceiro é o passo.

6. Integração NCL e Lua (NCLua)

Documentos NCL reconhecem scripts Lua como mídias, tal como imagens, vídeos (listagem 1.19).

```
<media id="lua" src="lua.lua" descriptor="dsLua"/>
```

Listagem 1.19. Definição de uma mídia NCLua

A integração entre objetos NCLua e documentos NCL se dá por meio do paradigma orientado a eventos. Para atender aos requisitos do paradigma, Lua precisou ser estendida para promover uma melhor integração. Além das bibliotecas padrão, foram adicionados os módulos:

- Módulo Event: permite que objetos NCLua se comuniquem com o documento NCL e outras entidades externas, tornando as aplicações orientadas a eventos.
- Módulo Canvas: oferece uma API que permite a alteração e manipulação de componentes gráficos.
- Módulo Settings: oferece acesso às variáveis definidas em um documento NCL e em variáveis reservadas de objetos do tipo “application/x-ncl-settings”.

- Módulo Persistent: permite que variáveis persistentes sejam acessadas e manipuladas por objetos imperativos.

Um script NCLua pode interagir com diversas entidades, tais como: eventos do controle remoto, eventos do canal de interatividade, documentos NCL, etc. Para ser informado quando eventos externos são recebidos, um NCLua deve registrar pelo menos uma função tratadora, que é registrada por meio da função event.register (listagem 1.20).

```
function handler(evt) -- código do tratador de eventos
        ...
end
event.register(handler) -- registro da função tratadora
```

Listagem 1.20. Registro de uma função tratadora de eventos

Eventos são definidos como tabelas Lua com chaves e valores descrevendo seus atributos. Na listagem 1.21 a tabela do evento indica que o botão vermelho foi pressionado, onde *type* pode ser *press* ou *release*, *key* é o valor da tecla do controle.

```
evt = {
        class = 'key',
        type = 'press',
        key = 'RED'
}
```

Listagem 1.21. Representação de um evento em NCLua

O módulo *event* define as seguintes classes de eventos:

- Classe ‘ncl’: usada na comunicação entre um NCLua e o documento NCL que contém o objeto de mídia;
- Classe ‘key’: representa o pressionamento de teclas do controle remoto do usuário;
- Classe ‘tcp’: permite acesso ao canal de interatividade por meio do protocolo TCP;
- Classe ‘sms’: usada para envio e recebimento de mensagens SMS em dispositivos móveis;
- Classe ‘edit’: permite que comandos de edição ao vivo sejam disparados a partir de scripts NCLua;
- Classe ‘si’: provê acesso a um conjunto de informações multiplexadas em um fluxo de transporte e transmitidas periodicamente por difusão;
- Classe ‘user’: através dessa classe, as aplicações podem estender sua funcionalidade criando seus próprios eventos.

Na norma ABNT NBR 15606-2:2007 [ABNT 2007], pode-se encontrar todas as funções do módulo *canvas* e mais informações sobre os módulos que ainda não foram implementados.

Referências

ABNT NBR 15606-2 (2007) – Associação Brasileira de Normas Técnicas, "Televisão digital terrestre – Codificação de dados e especificações de transmissão para radiodifusão digital – Parte 2: Ginga-NCL para receptores fixos e móveis – Linguagem de aplicação XML para codificação de aplicações", Sistema Brasileiro de TV Digital Terrestre, *NBR 15606-2*.

Barbosa, S. D. J., Soares, L. F. G., "TV Digital Interativa do Brasil se faz com Ginga: fundamentos, padrões, autoria declarativa e usabilidade".

Ierusalimschy, R. (2006). "Programming in Lua", Segunda Edição. http://lua.org.

NCL – Nested Context Language. http://www.ncl.org.br/, 2010.

Sant'Anna, F. F. G., Soares Neto, C. S., Azevedo, R. G. A. e Barbosa, S. D. J. "Desenvolvimento de Aplicações Declarativas para TV Digital no Middleware Ginga com objetos imperativos NCLua" .

Sant'Anna, F. F. G., Soares, L. F. G. e Cerqueira, R. F. G., Nested Context Language 3.0 Part 10 – Imperative Objects in NCL: The NCLua Objects, 2008.

Soares Neto, C. S. et al. Construindo Programas Audiovisuais Interativos utilizando a NCL 3.0 e a ferramenta Composer. 2007.

Soares, L. F. G., Rodrigues, R. F. Nested Context Model 3.0 Part 1 – NCM Core, 2005.

Soares, L. F. G. e Barbosa, S. D. J. (2009) Programando em NCL 3.0. São Paulo/SP. 1ª ed. 2009.

Cluster de serviços e alta disponibilidade com Software Livre

Patrick Meneses Melo
Faculdade de Tecnologia de Teresina – CET - Teresina-PI-Brasil
patrickimelo@hotmail.com

Abstract. *Dependence on computer systems became visible these days, and the possibility of unavailability of services provided by the systems became intolerable. Have you ever imagined the size of the damage to businesses if the system of stock market goes down? Or the payroll system of your company stop working on your payment? The goal of this project is to build a highly available clustered environment using free software.*

Resumo. *A dependência de sistemas computacionais se tornou visível nos dias de hoje, e a possibilidade de indisponibilidade dos serviços providos pelos sistemas se tornou intolerável. Já imaginou o tamanho do prejuízo para as empresas se o sistema da bolsa de valores sair do ar? Ou o sistema da folha de pagamento da sua empresa parar de funcionar no dia do seu pagamento? O objetivo desse projeto é construir um ambiente clusterizado e altamente disponível, utilizando software livre.*

1. Introdução

Um Cluster pode ser definido como um sistema que compreende dois ou mais computadores ou sistemas, denominados nodos ou nós, no qual trabalham em conjunto para executar aplicações ou realizar outras tarefas, de maneira transparente aos usuários, ou seja, os usuários terão a impressão de estar utilizando um único sistema [1].

HA (High availability) Alta disponibilidade - Técnica de projetar sistemas de forma que os serviços fornecidos por eles cumpram os requisitos de estarem acessíveis por um período mínimo definido por um contrato de nível de serviço ou SLA (Service Level Agreement).

2. Categorias

2.1. Sistemas Gerais

São aqueles em que falhas e interrupções curtas de serviço são toleráveis desde que o sistema volte a funcionar posteriormente.

2.2. Sistemas Altamente Disponíveis

Disponibilidade crítica, medida a maneira de se determinar a qualidade dos serviços, sendo inaceitável a falha do sistema como um todo. Ex. Sistemas bancários, telecomunicações.

2.3. Sistemas Computacionais Críticos

Podem comprometer a segurança de pessoas ou ter um alto impacto econômico. Exemplos: Sistema Aéreo, Militar, Industriais.

2.4. Sistemas de Longa Vida

A confiabilidade é prioridade máxima e normalmente é impossível realizar manutenção não planejada. Exemplos: Satélites, Controladores de Vôo Espacial.

Todo sistema possui uma taxa de confiabilidade, que é a probabilidade do mesmo realizar suas funções em dado período de tempo, utilizada para calcular o SLA. Bem comuns são o MTBF (Medium Time Between Failures) e o MTTR (Medium Time to Repair).

3. Downtime

Tempo que determinado serviço ou sistema ficou indisponível, podendo ocorrer de duas formas: planejada e não-planejada. Períodos não planejados são decorrentes de falhas não controladas e devem ser evitadas através de técnicas de alta disponibilidade. Períodos planejados são normalmente períodos de manutenção agendados de maneira a minimizar o impacto no negocio e tem o objetivo de adicionar, modificar, reparar ou atualizar componentes do sistema.

O SLA pode ser estimado conhecendo-se o MTBF dos sistemas utilizados aplicando a formula:

Disponibilidade (%)	Downtime Anual	Downtime Mensal	Downtime Semanal
90%	36,5 dias	72 horas	16,8 horas
99%	3,65 dias	7,2 horas	1,68 horas
99,9%	8,76 horas	43,2 minutos	10,1 minutos
99,9999%	31,5 segundos	2,59 segundos	0,6 segundos

Tabela 1. Cálculo de disponibilidade

3. Causas de Indisponibilidade

Pesquisas realizadas em 2010 apontam os maiores causadores de indisponibilidade:

3.1. Falta de boas praticas de gerencia de mudanças.

3.2. Falta de boas praticas de monitoração dos componentes dos sistemas.

3.3. Falta de boas praticas de definição de requisitos.

3.4. Falta de boas praticas de procedimentos operacionais.

3.5. Falta de boas praticas em evitar falhas de rede.

3.6. Falta de boas praticas em evitar problemas internos com aplicações.

3.7. Falta de boas praticas em evitar falha de serviços externos.

3.8. Ambiente físico inadequado.

3.9. Falta de boas praticas de modelo de redundância de rede.

3.10. Falta de solução de backup.

3.11. Falta de boas praticas na escolha da localização do Datacenter.

3.12. Falta de redundância de infra-estrutura.

3.13. Falta de redundância de arquitetura de Storage.

4. Modelos de Alta disponibilidade

Um ambiente que contem computadores redundantes ou Nós, é chamado de cluster. Seu principio de funcionamento é que caso um dos nós falhe, o outro nó assumirá o lugar do outro de forma transparente até que seja realizado o reparo. A utilização de redundância visa eliminar a necessidade de intervenção humana no ambiente para que o mesmo continue em funcionamento em caso de falhas.

4.1. Redundância Passiva

Nestes sistemas a alta disponibilidade é alcançada adicionando-se capacidade extra no projeto do sistema de forma que a falha de um componente reduza a performance do ambiente, porem o mesmo continue funcionando de maneira suficiente a atender seus objetivos.

4.2. Redundância Ativa

Nestes sistemas a alta disponibilidade é atingida sem perca de performance, e envolve a utilização de mecanismos mais complexos e capazes de detectar a falha de componentes e reconfigurar automaticamente o ambiente.

5. Classificação de Nós em um cluster

O tamanho mais comum em um cluster de alta disponibilidade é o cluster de 2 nós, pois é o mínimo necessário para fornecer redundância, porem existem outras configurações mais complexas.

5.1. Ativo/Passivo

Provê uma instancia completamente redundante do serviço em cada nó, que é ativada apenas em caso de falha no nó primário.

5.2. Ativo/Ativo

Os nós compartilham os dados e podem operar de forma conjunta. Em caso de falha o trafego destinado ao nó que falhou é redirecionado para algum dos nós remanescentes ou balanceado por um balanceador existente na "frente" do cluster.

5.3. N+1

Semelhante a configuração Ativo/Ativo, porem utiliza um nó extra, passivo, capaz de assumir todos os serviços de um nó que eventualmente falhe.

5.4. N+M

Em alguns clusters é possível que apenas um nó extra não seja capaz de fornecer redundância suficiente. Nestes casos mais de um nó são mantidos inativos, sendo ativados de acordo com necessidade.

6. Comunicação entre os Nós de um cluster

Para o pleno funcionamento é necessário haver alguma forma de coordenação entre os nós, para que cada um "saiba" quais serviços estão ativos e em que membro. Assim é possível que serviços sejam iniciados e parados por membros específicos.
Essa comunicação é feita através do Stack de Clusterização, que é composto de duas partes:

6.1. Gerenciador de Recursos de Cluster (CRM)

Responsável por iniciar e parar os serviços que o cluster mantém.

6.2. Mensageiro

É responsável pela comunicação entre os Nós do cluster, mantendo atualizadas as informações de processos e estados dos servidores.

7. LVS (Linux Virtual Server)

É uma solução de balanceamento de carga avançada para sistemas Linux. É um projeto Open Source começado por Wensong Zhang em maio de 1998. A missão do projeto é construir um servidor de alto desempenho e altamente disponível para Linux usando a tecnologia de clustering. Servidor virtual é um servidor altamente escalável e altamente disponível construído em um cluster de servidores reais. A arquitetura de cluster de servidor é totalmente transparente para os usuários finais, e os usuários interagem com o sistema de cluster, como se fosse apenas um servidor de alta performance virtual único.

O suporte ao LVS é parte do Kernel e sua parte principal é o código do ip_vs (IP Virtual Server), executado em um servidor denominado diretor do grupo LVS. Este diretor é implementado através do daemon *ldirectord*. O diretor age como um switch de camada 4, recebendo solicitações de conexão de um cliente e escolhendo um servidor para atender a solicitação. Um servidor backend é denominado *realserver* na terminologia do LVS. O LVS será utilizado para balancear a carga dos acessos entre os

servidores reais presentes no cluster, e caso todos os servidores reais fiquem indisponíveis o serviço será automaticamente migrado para o host do próprio LVS.

8. Ambiente Proposto

Abaixo poderemos visualizar como ficará nosso ambiente clusterizado com os servidores reais, para isso utilizaremos o modelo LVS (NAT). A topologia NAT permite uma grande latitude na utilização de hardware existente, mas é limitado em sua capacidade de lidar com grandes cargas devido ao fato de que todos os pacotes passarão pelo roteador LVS.

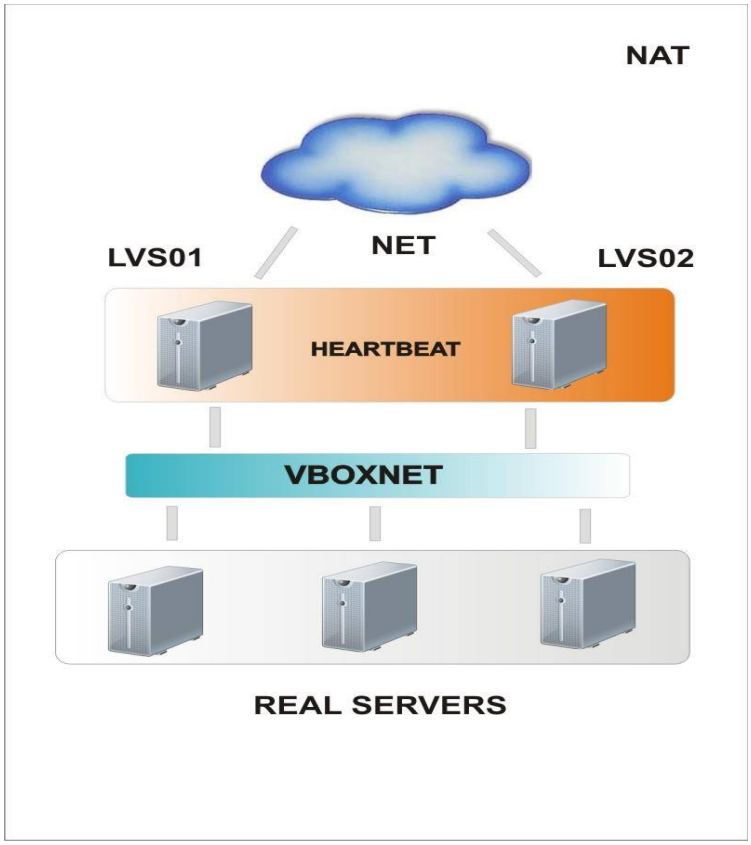


Figura 1. Ambiente proposto

8.1. Instalação do Ldirectord

Primeiro passo na montagem do LVS é instalar os aplicativos necessários para controlar o Diretor. Isso é feito, no debian, instalando o pacote do diretor LVS.

 # apt-get install ldirectord

Copie o arquivo de configuração que está no diretório HOME do usuário para o diretório padrão do ldirectord.

 # cp /home/files/ldirectord.cf /etc/ha.d/

Existe um arquivo de exemplo presente na documentação do ldirectord, para visualizá-lo execute:

 # zcat/usr/share/doc/ldirectord/examples/ldirectord.cf.gz

Por fim vamos apontar a localização correta do arquivo de configuração no script de inicialização do ldirectord, edite o arquivo" /etc/init.d/ldirectord " na linha**:**

CONFIG_FILE="${CONFIG_FILE:=/etc/ldirectord.cf}"

Para:

CONFIG_FILE="${CONFIG_FILE:=/etc/ha.d/ldirectord.cf}"

No momento da elaboração do curso, a ultima versão disponível do ldirectord é a v1.186-ha, e possui um bug documentado que fazia com que o daemon terminasse inesperadamente quando havia um timeout da verificação do HTTP, aplique o patch para corrigir:

 # patch –p0 /usr/sbin/ldirectord /root/LVS/ldirectord.patch

8.2. Configuração do ldirectord

O arquivo que copiamos anteriormente é um modelo que deve ser adequado de acordo com o ambiente proposto, visualize o arquivo /etc/ha.d/ldirectord.cf.

```
# Tempo sem resposta, em segundos, até declarar um servidor fora do ar.
checktimeout=3
# Intervalo de verificação dos servidores, em segundos
checkinterval=1
# Recarregar o serviço em caso de alteração no arquivo de configuração
autoreload=yes
# Não remover da tabela de roteamento servidores que falharem, apenas definir seu peso como "0"
quiescent=yes
# Definição de um novo serviço do Cluster
virtual=ip.ip.ip.ip:80
        real=10.240.0.20:80 masq
        real=10.240.0.21:80 masq
        real=10.240.1.20:80 masq
        real=10.240.1.21:80 masq
        fallback=127.0.0.1:80 gate
        fallbackcommand="/etc/init.d/apache2"
        service=http
        request=".testpage"
        receive="test page"
```

scheduler=rr
protocol=tcp
checktype=negotiate

Uma explicação mais detalhada de cada diretiva de serviço:

8.3. Virtual

Define um serviço indicado por endereço IP, porta e/ou marca de firewall.

8.4. Real

Define o realserver que irá atender requisições deste servidor virtual.Note que a diretiva "**masq**" após o endereço, especifica que será realizado mascaramento de endereço.

8.5. Fallback

Endereço para o qual serão enviadas as solicitações caso haja falha em todos os Realservers. Normalmente utilizada como feedback para o cliente.

8.6. FallbackCommand

Comando que sera executado com parametro "start" caso todos os realservers estejam indisponiveis e com o parametro "stop" quando o primeiro realserver se tornar disponivel.

9. Service

Tipo de serviço que esta sendo verificado. Tipos validos são: dns, ftp, http, https, http_proxy, imap, imaps, ldap, mysql, nntp, Oracle, pgsql, pop, pops, radius, simpletcp, sip, smtp, para serviços genéricos utilizamos o "simpletcp".

9.1. Request

URI do objeto que será solicitado para verificação.

9.2. Receive

Resposta (conteudo) que é esperada apos solicitor o objeto na solicitação acima.

10. Scheduler

Forma que a carga será distribuída entre os realservers do cluster. Alguns dos agendadores possíveis são:

10.1. RR

Round Robin, distribui conexões igualmente entre os servidores a medida que são feitas.

10.2. WRR

Weighted Round Robin, designa conexões para servidores de acordo com pesos (prioridades).

10.3. LC

Least Connnections, envia novas requisições para servidores que estão no momento com pouca demanda.

10.4. WLC

Weighted Least Connnections, envia novas requisições para servidores que estão no momento com pouca demanda, de acordo com pesos definidos.

11. Checktype

Tipo de verificação que será efetuada. Os tipos são:

11.1. Negociate

Irá tentar uma conexão utilizando o protocolo definido em "Service" e tentar obter uma resposta válida para uma solicitação.

12. Connect

Irá apenas tentar abrir uma conexão na porta indicada no servidor virtual.

12.1. Ping

Irá utilizar o protocolo ICMP para verificar a disponibilidade dos realservers.

13. Ajustes necessários:

Será necessário editarmos algumas configurações referentes a rede das VM's, para que não haja conflito nos endereçamentos.

14. Validando LVS:

Para testarmos o funcionamento do diretor LVS devemos em primeiro lugar iniciar o diretor, execute:

 # /etc/init.d/ldirectord start

Os Logs relatives ao funcionamento são armazenados por padrão em /var/log/ldirectord.log, visualize:

 # tail –f /var/log/ldirectord.log

14.1. Redundância para o LVS

Apesar de termos os serviços balanceados entre vários servidores, devemos criar a redundancia para evitarmos o ponto de falha de um único servidor LVS, para isso utilizaremos o Heartbeat para configurar a clusterização.

15. Heartbeat

O Sistema de Alta Disponibilidade com maior maturidade e mais utilizado no sistema operacional Linux é o Heartbeat [2]. Ferramenta que funcionará como gerente do cluster e pode ser chamado de o "coração" da infra-estrutura de alta disponibilidade, é o daemon responsavel pela comunicação entre os Nós do cluster.

15.1. Instalação do Heartbeat

Para a instalação iremos utilizar o repositorio:

apt-get install heartbeat

Apos a instalação vamos copiar os arquivos de configuração criados em "/root/heartbeat/".

cp /root/heartbeat/* /etc/ha.d/

Para que funcione a comunicação entre os Nós devemos criar credencial para autenticação. Adicione o conteudo no arquivo "/etc/ha.d/authkeys"

auth 1

1 md5 123456

O arquivo de configuração principal do heartbeat é o "ha.cf" e o significado das diretivas são:

15.2. Keepalive

Intervalo entre o envio de pacotes de verificação do Heartbeat.

15.3. Deadtime

Tempo para que o Heartbeat leva para informar que um nó está inativo depois da verificação.

15.4. Warntime

Tempo que o heartbeat leva para emitir um aviso de demora para a resposta do pacote keepalive.

15.5. Initdead

Semelhante ao Deadtime, porem é utilizado na inicialização do serviço e em valores maiores, permitindo o inicio em todos os nós.

15.6. Bcast

interface na qual o Heartbeat utilizará para o broadcast dos pacotes de verificação.

15.7. Node

Informa quais maquinas farão parte do cluster, é importante que a maquina seja capaz de resolver nomes e seu próprio hostname conste nela.

15.8. Crm

Especifica se será utilizado um gerenciador de recursos externo.

15.9. Auto_failback

Especifica se um recurso deverá retornar ao nó primário após uma recuperação de falha.

15.10. Debug e Logfile

Definem o arquivo que será utilizado para gravar as informações de log do serviço.

Para seu efetivo funcionamento devemos informar qual recurso será compartilhado entre os servidores, antes de tudo devemos configurar o VIP (ip virtual) utilizado pelo cluster. Edite o arquivo "/etc/ha.d/haresources"

lvs01 VIP ldirectord

Dessa forma o Heartbeat irá gerenciar o VIP(Ip Virtual), e o ldirectord alem de informar que o nó primário será o lvs01, onde deverão ser mantidos os serviços a menos que esteja indisponível.

16. Testando as configurações

No momento já possuímos uma estrutura básica montada, o Heartbeat irá gerenciar o LVS nos nós. Para os testes devemos adicionar o VIP no LVS e parar o serviço de LVS. Edite o arquivo "/etc/ha.d/ldirectord.cf" e modifique para o ip do VIP:

virtual=ip.ip.ip.ip

Pare o serviço do LVS:

/etc/init.d/ldirectord stopI

Inicie o serviço do Heartbeat:

/etc/init.d/heartbeat start

Vamos verificar os arquivos de log do Heartbeat para verificar se os serviços irão subir corretamente:

tail -f /var/log/ha-debug.log

Check se o serviço do LVS e o IP estão definidos no host:

service ldirectord status & ifconfig eth1:0

16.1. Adicionando o segundo Nó

A configuração e instalação do Heartbeat será a mesma para o nó primario, após a instalação é necessario copiar a chave de autenticação entre os nós:

scp -p /etc/ha.d/authkeys root@lvs02:/etc/ha.d/

Inicie o Heartbeat no segundo nó e verifique o log de erros:

tail -f /var/log/ha-debug.log

16.2. Testes de funcionamento do cluster

Para testar a redundancia vamos parar o Heartbeat no nó primario e verificar se os serviços migrarão para o nó secundário.

17. Conclusão

Neste mini-curso conseguimos em um curto espaço de tempo criar um balanceamento de carga de serviços entre servidores e também configurar a alta disponibilidade dos mesmos, como sempre, não existe uma fórmula mágica para calcular o ponto ideal (é justamente por isso que existem consultores e analistas), mas é sempre prudente ter pelo menos um nível mínimo de redundância, nem que seja apenas um backup atualizado, que permita restaurar o servidor (usando outra máquina) caso alguma tragédia aconteça.

Referências

[1] Sinhoreli, Marcos. (2008) “Armazenamento com alta disponibilidade: Os dados não param”, Linux Magazine - 43ª edição.

[2] Linux-HA. “Heartbeat”, http://www.linux-ha.org/Heartbeat

http://www.linuxvirtualserver.org/VS-NAT.html

http://www.austintek.com/LVS/LVS-HOWTO/HOWTO/LVS-HOWTO.LVS-NAT.html

https://access.redhat.com/knowledge/docs/pt-BR/Red_Hat_Enterprise_Linux/5/html/Virtual_Server_Administration/ch-lvs-setup-VSA.html

http://www.vivaolinux.com.br/artigo/Alta-disponibilidade-com-Debian-Lenny-+-Heartbeat-+-DRBD8-+-OCFS2-+-MONIT-+-LVS

Criação de projetos 3D com software Blender

Caio Farias Bitterncourt[1,]

[1] Universidade Estadual do Piauí (UESPI)
Caixa Postal 64200-000 – Parnaíba – PI – Brasil

caicarybio@gmail.com

Abstract. *The proposed work aims to present concepts essential for the production of models and designs in 3D using the open source tool Blender, given that this area of computing is being constantly used as a "framework" in both segments is market areas or science. It was chosen for this software is extremely feature rich 3D production with the advantages of being a free software and open source platform, and has a comprehensive community dedicated to it, which will show their origin, their environment and basic tools for the creation of projects.*

Resumo. *O trabalho proposto tem como objetivo apresentar conceitos primordiais para a produção de modelos e projetos em 3D utilizando a ferramenta de software livre Blender, em virtude que esta área da computação esta sendo constantemente usada como "framework" em ambos os segmentos seja áreas do mercado ou da ciência. Para isso fora escolhido esse software que é extremamente rico em recursos de produção 3D com as vantagens de ser um software gratuito de código aberto e multiplataforma, além de possuir uma abrangente comunidade dedicada ao mesmo, onde será mostrada sua origem, seu ambiente e ferramentas básicas para a criação dos projetos.*

1. Introdução

A computação gráfica pode ser entendida como o conjunto de algoritmos, técnicas e metodologias para o tratamento e a representação gráfica de informações através da criação, armazenamento e manipulação de objetos, utilizando-se computadores e periféricos gráficos. Atualmente com a integração das diversas técnicas através da tecnologia da informação o uso da computação gráfica possui um grande papel nas seguintes áreas: (CAD,Apresentações Gráficas, CGI, Artes por Computador, Entretenimento, Educação e Treinamento, Visualização Científica).

Neste capítulo, apresentaremos um histórico sobre o surgimento e evolução da computação gráfica, discutiremos também sobre a área de modelagem 3D falando sobre os conceito básicos de abstração para a modelagem, um breve histórico da criação do Blender com suas características e ferramentas chaves e por fim serão mostrados passos e ações básicas para modelagem de objetos em três dimensões.

2. História da Computação Gráfica

Pode-se dizer que a história da computação gráfica iniciou-se há tempos remotos, desde quando o homem começou a utilizar cálculos matemáticos para definir figuras geométricas. Todos os avanços matemáticos possibilitaram o surgimento do computador e da computação gráfica. Desde então os pesquisadores são atraídos pelo fascínio de poder observar graficamente informações digitais. Esse processo, que pode ser entendido como um modo de visualizar os dados na memória do computador que constitui hoje a área conhecida por Computação Gráfica que fora evoluindo em paralelo com avanços e surgimento de novas tecnologias do qual possibilitaram e deram impulso ao desenvolvimento desta área.

Com o avanços dessas tecnologias nos anos 50 surgiram os primeiros computadores com capacidades gráficas. No entanto a capacidade gráfica daquela época se resumia a apresentar pequenos pontos, símbolos ou números em um tubo de raios catódicos, sendo assim ainda não se podiam criar gráficos ou desenhos, mais apesar disso a possibilidade de poder ver resultados de informações em uma tela, representou um grande salto tecnológico.

O primeiro computador a possuir recursos gráficos de visualização de dados e processamento em tempo real foi o "Whirlwind I" (furacão), desenvolvido pelo MIT (Massachusetts Institute of Technology) com esse computador o comando de defesa aérea dos EUA desenvolveu um sistema de monitoramento e controle de vôos (SAGE - Semi-Automatic Ground Enviroment) que convertia as informações capturadas pelo radar em imagem em um tubo de raios catódicos.

Outros dois grandes marcos foram em 1961 no MIT onde surgiu o primeiro jogo de computador (Spacewars) para o computador DEC PDP-1 e logo após em 1963 que ocorreram as primeiras implementações de desenhos para um display de computador definitivo, criado por Ivan E. Sutherland em sua tese de doutorado intitulada de Sketchpad, esse display tinha a capacidade de reorganizar as leis físicas por meio da óptica e até mesmo de visualizá-las através de matéria computadorizada.

Essa tecnologia chamou a atenção das indústrias automobilísticas e aeroespaciais americanas que a adoram como ferramenta e desenvolveram um dos precursores dos primeiros programas de CAD(Computer-Aided Design ou desenho assistido por computador) desenvolvido pela General Motors chamado de CAD DAC-1, logo após no final da década de 60 praticamente toda a indústria automobilística e aeroespacial se utilizava de softwares de CAD e a partir de então começou uma grande atividade na investigação fundamental da computação gráfica surgindo e aprimorando algoritmos, métodos e técnicas que propulsionaram a computação gráfica a ganhar espaço em outras áreas.

No final da década de 60 surgi a primeira matriz de pixels, a computação Gráfica 2D desenvolve-se muito rapidamente e aparecem algoritmos fundamentais eficientes e ao inicia da década de 70 vários pesquisadores desenvolveram novas técnicas e algoritmos que são utilizados até hoje, tais como os métodos de sombreamento e o algoritmo de z-buffer. Na década de 1970, vários pesquisadores desenvolveram novas técnicas e algoritmos. Nessa mesma época, surgiu à tecnologia dos circuitos integrados permitindo o barateamento das máquinas e o lançamento, em 1975, do primeiro computador com interface visual, o predecessor do Macintosh.

Nesse mesmo ano Nolan Bushnell funda a empresa ATARI e lança o vídeo game Pong. Outros fatos importantes dessa década foram o reconhecimento da computação gráfica como área específica da ciência da computação, o surgimento dos congressos específicos em computação gráfica(SIGGRAPH), a publicação do primeiro livro sobre computação gráfica interativa e o lançamento em 1977 do livro Fractals: Form, Chanceand Dimension, onde o autor, Benoit Mandelbrot, matemático e, na época, pesquisador da IBM, conseguiu mostrar com imagens geradas em computador a incrível complexidade das equações fractais.

Esses avanços deram início à criação de curtas animações para fins didáticos e publicitários e a indústria de entretenimento começou a olhar com interesse para esta área em suas produções como o pioneiro Star Wars.

Em 1980 Loren Carpenter mostra no SIGGRAPH animações 3D realistas com paisagens geradas por métodos fractais e 1983 a Disney usa essas técnicas na produção do filme Tron que fora o primeiro filme a utilizar computação gráfica de forma efetiva.

Em 1986 S. Jobs compra a Pixar da empresa LucasFilm A INTEL e a Texas Inst. desenvolvem processadores gráficos e em 1988 a Pixar tem o filme Luxo Jr. nomeado para um Oscar e recebe a patente do programa de síntese de imagem RENDERMAN.

No final deda década de 80 e inicio da de 90 surge a linguagem de programação Open GL em 1992 e com isso uma infinidade de aplicações e filmes baseados em computador. O mercado da computação gráfica atinge seu estágio de maturidade apresentando um grande crescimento com produções realistas e técnicas avançadas de iluminação e modelagem. São exploradas outras possibilidades de geometrias além do espaço tridimensional, que são utilizadas com uma frequência cada vez maior pelas pessoas que trabalham com artes, computação e visualização científica.

3. Conceitos Básicos para Modelagem 3D

Quando pensamos em computação gráfica nos dias atuais associamos a filmes de animação 3D, jogos virtuais, efeitos especiais etc. Mais o que foi citado acima é apenas uma subárea da mesma cujo define aparte de modelagem e produção 2D ou 3D

Esse segmento da computação gráfica pode ser encarado como uma ferramenta não convencional que permite ao artista transcender das técnicas tradicionais de desenho ou modelagem que exigiriam do artista o uso de uma técnica apurada de desenho podendo ser geradas mais facilmente com o auxílio de softwares aplicando as ferramentas que eles oferecem, mais para isso há a necessidade de abstrair esses modelos do espaço real para o ambiente computacional, pois nosso mundo é extremamente complexo sendo preciso converte-lo para o ambiente computacional de forma a se criar ferramentas de simplificação de realidades infinitas em contextos discretos.

Esse conceito de simplificação é crucial para esta área como também da computação gráfica como todo, pois, como exemplo, sabemos que no em nosso espaço real existe infinitos números entre um determinado intervalo já esse conceito no ambiente computacional é filtrado para que seja aceitável.

A imagem abaixo ilustra o exemplo:

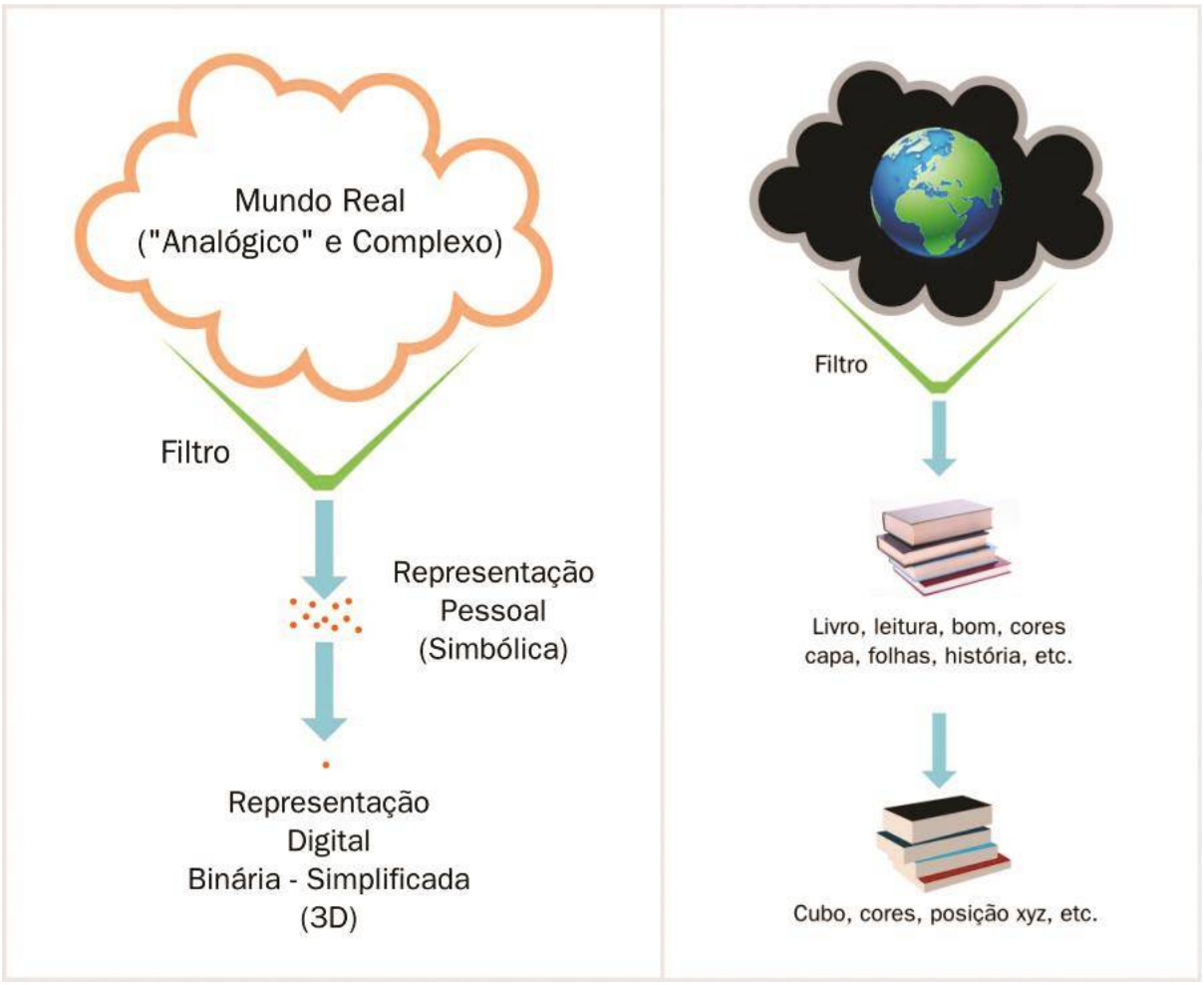


Figura 1. Representação da filtragem do Ambiente real para o ambiente computacional.

Para construir modelos 3D no ambiente computacional temos primeiro imagina-los de formas simplificada para as ferramentas limitadas que os programas nos oferecem como no exemplo abaixo:

Figura 2. Exemplo de simplificação do mundo real para o ambiente

Outro fator importante para adentrar no mundo computacional seja 2D ou 3D é ter conhecimento sobre o sistema de coordenadas do plano cartesiano, pois as imagens são formadas de (n) cruzamento de (n) valores seja de duas ou três dimensões onde para os objetos em 2D esse plano cartesiano será composto por dois eixos sendo um disposto horizontalmente e o outro verticalmente.

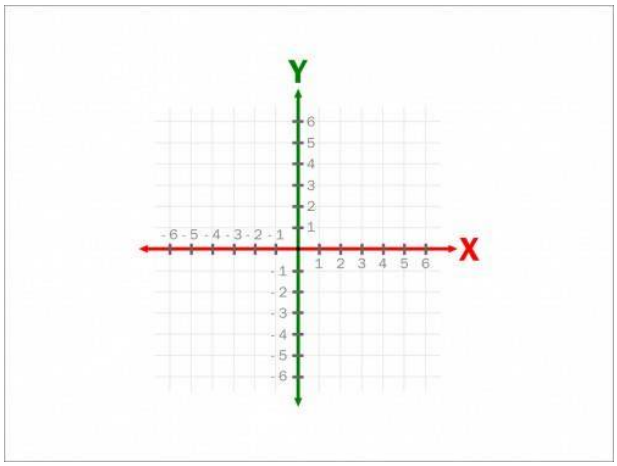


Figura 3. Representação do plano cartesiano de duas dimensões.

Com essa informação, podemos achar a posição de um ponto em qualquer parte do espaço, tendo a origem como referência.

Já nos tratando de objetos com 3D acrescentamos mais um eixo em nosso plano cartesiano

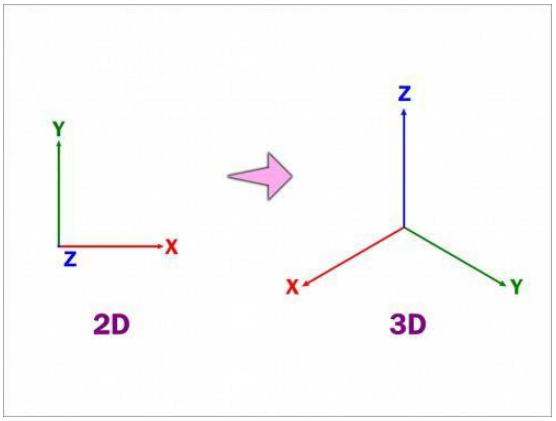


Figura 4. Representação do plano cartesiano de três dimensões.

Como o objetivo desse trabalho é a produção de modelos em 3 dimensões adotaremos o plano cartesiano de três eixos para a construção de nossos modelos.

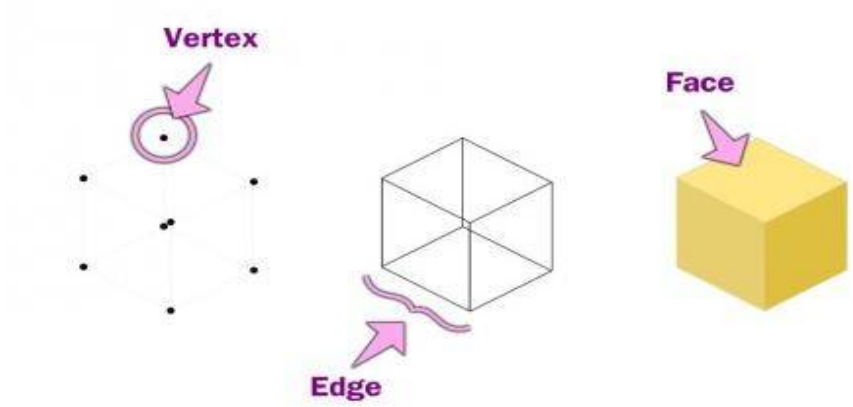


Figura 5. Vértices, arestas e faces de um objeto 3D.

Além disso para se poder criar os modelos em 3D precisamos dominar os três bloco de sólidos básicos que são: os vértices(vertex), as arestas(edge) e as faces(face)

Os vertices são os pontos do espaço tridemensional (X,Y e Z) eles servem como indicadores das arestas que é a ligação entre esse dois pontos e a união de no mínimo 3 arestas fomam uma face.

O conjunto dessas faces interligadas nos diferentes eixos(X,Y e Z) constituem um objeto de três dimensões e mudando seus valores ou acrescenanto mais desses blocos (vertices, artesas e faces) o objeto assumirá a forma desejada.

Dessa forma o processo de modelagem se torna mais demorado então por isso temos que fazer a abstração do mundo real para as formas da geometria que facilitaram o processo de modelagem. Pensando nisso todos os programas de modelagem existentes no mercado possuem as primitivas já prontas e editáveis que serviram de base para a construção dos modelos.

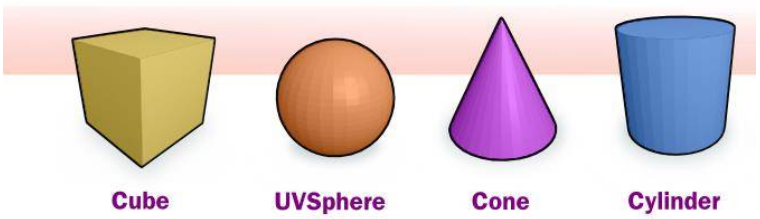


Figura 6. Formas geométricas primitivas.

Para se trabalhar com essas primitivas existem uma série de modificações que podem ser aplicados aos blocos que compõem o objeto como, por exemplo: Rotação, Expulsão, Escala, Bisel, Alise, Subdivisão, Fundição.

Com esses modificadores básicos podemos criar modelos já de níveis bem detalhados como segue abaixo a aplicação dos conceitos já mostrados e a utilização dos modificadores Expulsão, Escala, Subdivisão e Alise para a construção em 28 passos de um objeto do mundo real.

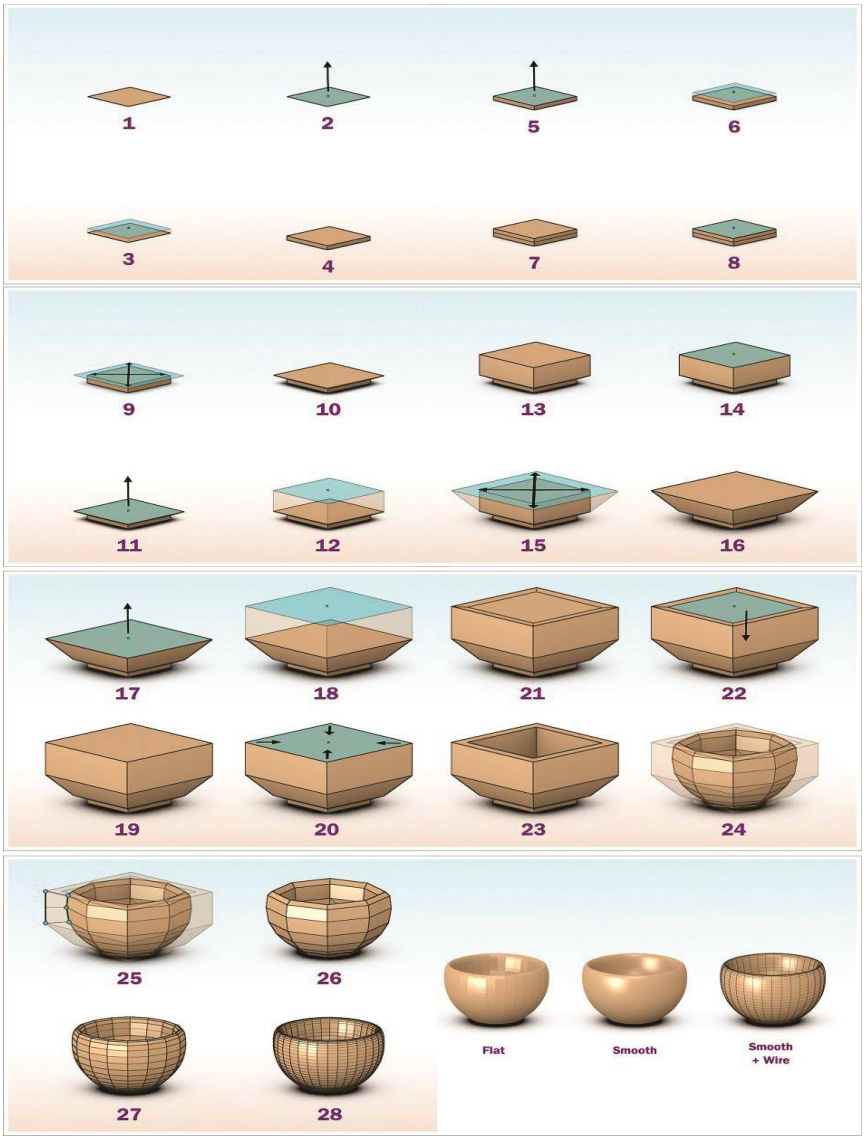


Figura 7. Construção de um modelo a partir de alguns modificadores básicos.

4. A Ferramenta Blender

4.1. Histórico

Blender é um programa de computador de código aberto, desenvolvido pela Blender Foundation, para modelagem, animação, texturização, composição, renderização, edição de vídeo e criação de aplicações interativas em 3D, tais como jogos, apresentações e outros, através de seu motor de jogo integrado, o Blender Game Engine. Está disponível sob uma licença dupla: Blender License (BL) / GNU General Public License (GPL).

Figura 8. Logomarca da Fundação Blender.

Possui ferramentas similares às de outros programas proprietários, que incluem avançadas ferramentas de simulação, tais como: dinâmica de corpo rígido, dinâmica de corpo macio e dinâmica de fluidos, ferramentas de modelagem baseadas em modificadores, ferramentas de animação de personagens, um sistema de composição baseado em "nós" de texturas, cenas e imagens, e um editor de imagem e vídeo, com suporte a pós-produção. Inclui suporte a Python como linguagem de script, que pode ser usada tanto no Blender, quanto em seu motor de jogo. Atualmente, suporta 25 idiomas, incluindo o português brasileiro.

Figura 9. Logomarca da ferramenta Blender.

E um programa multiplataforma, portanto disponível para diversos sistemas operacionais nas arquiteturas de 32 e 64 bits.

Originalmente, o Blender foi desenvolvido como uma aplicação in-house pelo estúdio holandês de animação NeoGeo Studio, co-fundado por Ton Roosendaal em 1988. Após isso Em 2002, a NaN faliu devido a pouca quantidade de vendas e a problemas financeiros. No mesmo ano, Ton fundou a Blender Foundation e em julho desse ano, iniciou-se uma campanha chamada "Free Blender", para arrecadar fundos para manter o projeto mais sendo um software livre.

4.2. Características

O Blender pode ser utilizado em qualquer área que seja necessária a geração de modelos tridimensionais, geração de imagens renderizadas, animação e jogos,4 como aplicações em arquitetura,14 design industrial, engenharia, animação, produção de vídeo, e desenvolvimento de jogos, graças ao seu motor de jogo embutido. Esta característica pode ser ampliada e agilizada com o uso de scripts em Python. Como modelador, foi recomendado pela Peugeot, para ser usado em seus concursos de design de carros, o Peugeot Design Contest.

Agregasse a ele também uma ferramenta chamada Sculpt, que possibilita trabalhar com modelos como se estivesse os esculpindo, semelhantemente ao modelador ZBrush.

Um de seus grandes diferenciais é o seu motor de jogo (Bender Game Engine), também conhecido como BGE, Game Blender ou Ketsji. Ele usa OpenGL para os gráficos, OpenAL para som 3D, Bullet para física e detecção de colisão, e Python para scripts. Existe um plugin, chamado Echo Plugin, que permite integração dos gráficos do OGRE(motor gráfico 3D orientado a objetos) com o Blender Game Engine. O uso do motor de jogo do Blender pode servir para diversas coisas, desde criação de jogos, apresentações, realidades virtuais, planejamento arquitetônico, a auxílio em animação

usando a física para dar movimentos mais reais aos objetos.

Cada material pode ser criado de modo procedural ou salvo em scripts. Estando em um arquivo separado, seu uso se torna mais flexível, pois existe herança de materiais. Esta herança também ajuda a diminuir a complexidade, ocultando operações redundantes.

4.3. Instalação

- ✓ A instalação do Blender é muito simples em ambas as plataformas vá até o seguinte site http://www.blender.org/download/;
- ✓ Escolha a versão de download para seu sistema operacional;
- ✓ Após isso siga os passos de instaçação e pronto;

5. Interações Básicas com a Ferramenta Blender

5.1. Comandos de rotação básicos

Assim que o Blender abrir, aparecerá uma tela inicial com a presente versão. Apenas clique com o botão esquerdo do mouse sobre a interface dele.

Após isso apresentará a seguinte tela abaixo com os seguintes menus a as seguintes ferramentas de acesso rápido.

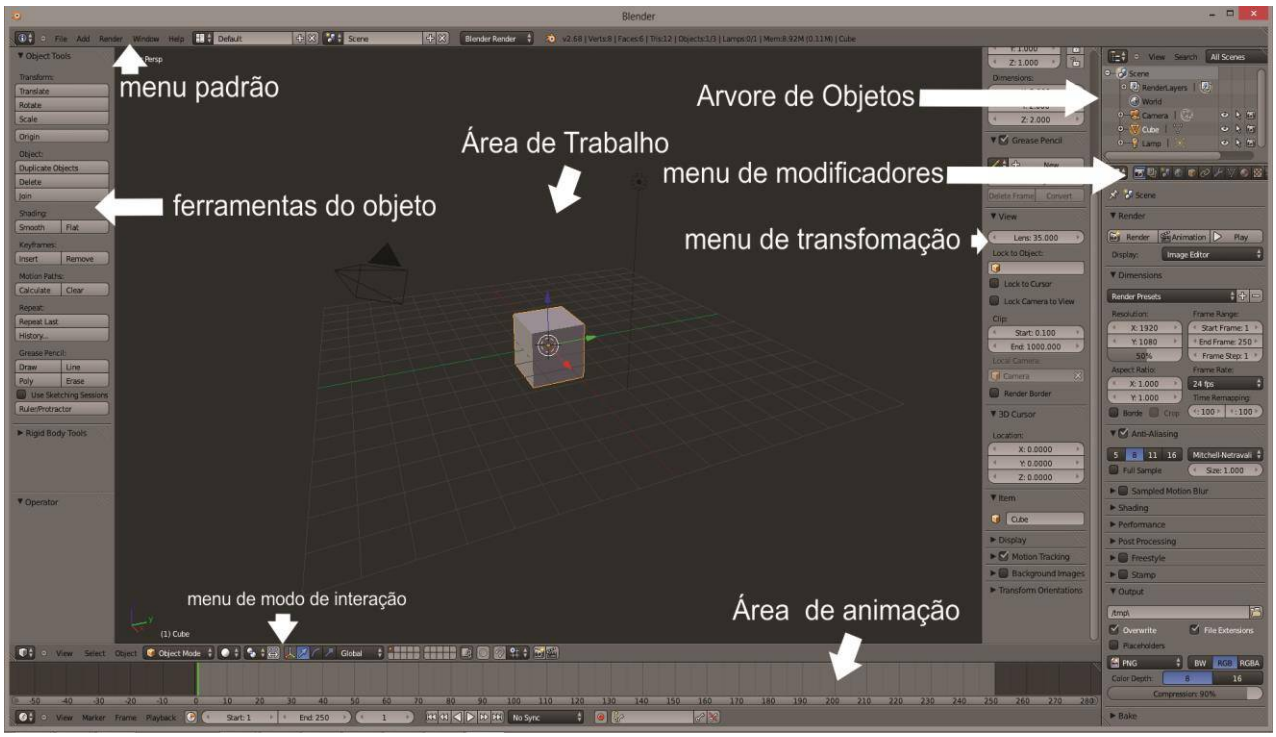


Figura 10. Ambiente Padrão de Trabalho do Blender.

Para rotacionar o objeto (cubo) mova a curso do mouse em qualquer área do cubo, clique no scroll do mouse sem soltá-lo, mova o mouse aos poucos para fazer a rotação. Ao chegar à visão desejada pode basta soltar o botão esquerdo do mouse.

Figura 11. Passos básicos de rotação do objeto.

5.2. Comandos de seleção básicos

Um objeto selecionado é aquele que tem uma linha alaranjada em torno de si. Observe o cubo, se ele tem uma linha alaranjada contornando-o, significa que está selecionado. Caso não tenha, é necessário selecioná-lo.

Para selecionar o objeto, mova o cursor do mouse mais ou menos ao centro do objeto e clique no botão direito do mouse e clique fora do cubo com o botão direto do mouse para retirar a seleção.

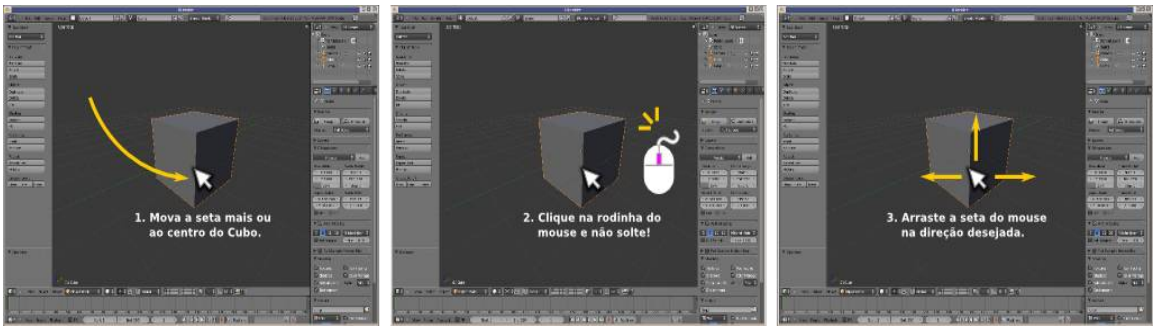


Figura 12. Passos básicos de seleção.

5.3. Comandos de movimentação básicos

Para mover um objeto é preciso selecionar o objeto, arrastando o curso do mouse para a área do objeto.

Uma vez que esteja ao centro clique no botão direito do mouse, ao fazer isso a linha alaranjada aparece em volta do objeto e os eixos ao centro.

Escolha o eixo de preferencia para que ele se mova a partir do eixo escolhido.

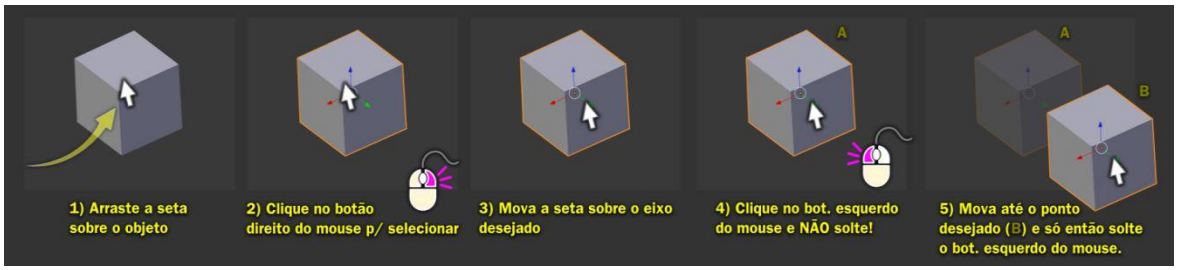


Figura 13. Passos básicos de movimentação.

5.4. Amentando Objetos

Para definir os tamanhos dos blocos selecionados seguem-se os seguintes passos:

Selecione o objeto, pressione o botão "S" e quiser diminuir mova a seta para o centro do objeto, se quiser aumentar, mova para fora do objeto até chegar ao tamanho desejado.

Ao chegar ao tamanho pressione o botão esquerdo do mouse.

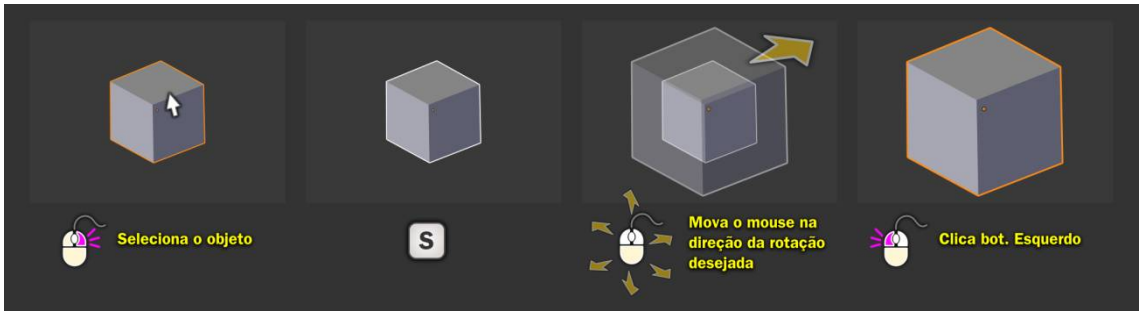


Figura 14. Acessando as formas primitivas.

5.5. Acessando formas primitivas

Para a seleção das demais formas primitivas existentes pressiona-se as seguintes teclas "SHIT + A" onde aparecera o menu de primitivas bastando clicar no que se deseja.

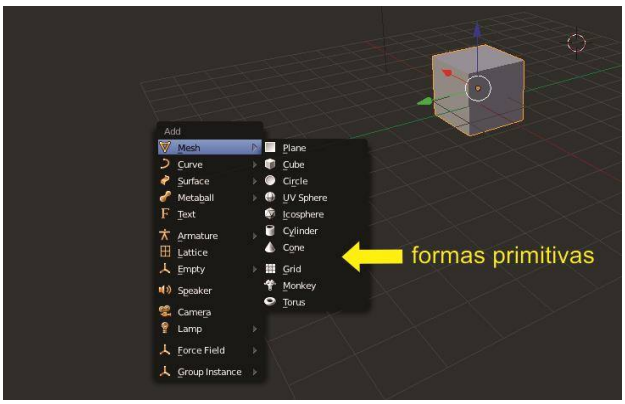


Figura 15. Menu de formas primitivas.

5.6. Alternando entre os modos de edição

Para alternar para o modo de edição dos blocos pressione a tecla “TAB” e após isso “CTRL + TAB” para definir qual bloco deseja modificar.

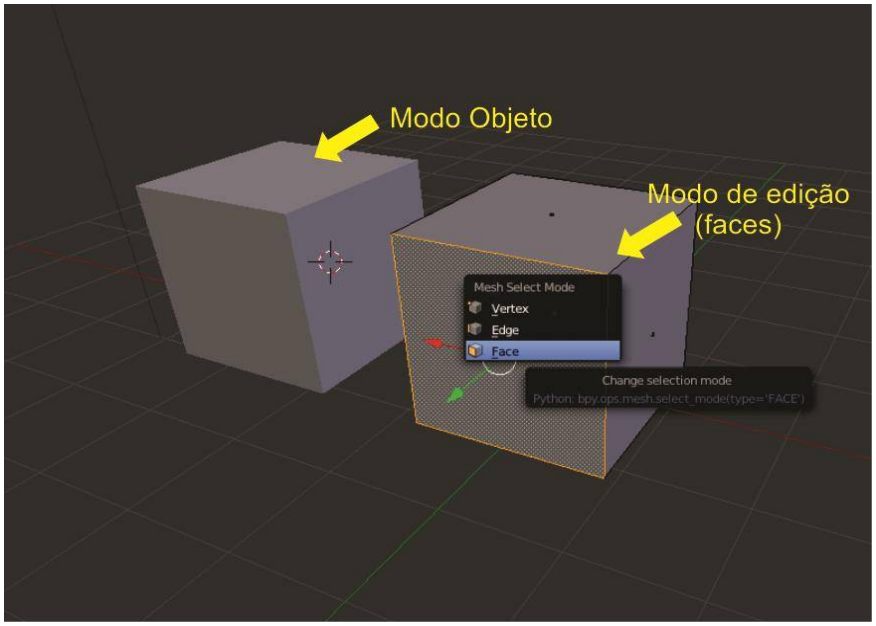


Figura 16. Modificando os blocos.

6. Referências

Luiz VELHO (2000). “Computação Gáfrica e Esilos Visuais”, http://www.visgraf.impa.br/Data/RefBib/PS_PDF/eav00/

Brito, Allan. “Blender 3D: guia do usuário / Allan Brito”, -- 4. ed. rev. e ampl. -- São Paulo : Novatec Editora, 2010.

Cícero Morais. “Blender 3D 2.5: tutorial para iniciantes”, http://www.hardware.com.br/tutoriais/novo-blender3d-iniciantes/

Robson Guedes. “História da Computação Gáfica”, http://fortium.edu.br/blog/robson_guedes/files/2010/03/Computa%C3%A7%C3%A3o-gr%C3%A1fica-Hist%C3%B3ria.pdf

Sevo, Daniel E. “History of Computer Graphics”, http://hem.passagen.se/des/hocg/hocg_1960.htm

Introdução à Engenharia de Software Orientada a Agentes com JaCaMo

Nécio L. Veras[1], Anderson C. P. Queiroz[2], Francisco R. O. Lima[2], Robert M. R. Júnior[2], Igor B. Nogueira[2], Mariela Inés Cortés[2], Gustavo A. L. Campos[2]

GESSI - Grupo de Engenharia de Software e Sistemas Inteligentes

[1]Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE)
Rodovia CE-187 – 62320-000 – Tianguá – CE – Brasil

[2]Universidade Estadual do Ceará (UECE) – Fortaleza, CE – Brasil

necio.veras@ifce.edu.br,{andersoncpdq,us.robson7,robstermarinho,
igor.bnog}@gmail.com,{mariela, gustavo}@larces.uece.br

Abstract. *This paper approaches an introduction to Agent-Oriented Software Engineering to motivate studies on the development of intelligent agents using three different technologies for developing multi-agent systems. Concepts, foundations and definitions related to multi-agent systems are presented in theory and practice through the illustration of two case studies for teaching purposes built using frameworks Jason and Cartago.*

Resumo. *Este trabalho aborda uma introdução à Engenharia de Software Orientada a Agentes para motivar estudos sobre o desenvolvimento de agentes inteligentes por meio de três diferentes tecnologias de desenvolvimento de sistemas multi-agentes. Conceitos, fundamentos e definições relacionados aos sistemas multi-agentes são apresentados de forma teórica e prática através da ilustração de dois estudos de casos construídos para fins didáticos usando os frameworks Jason e Cartago.*

1. Introdução

Cada vez mais, requisitos relacionados à utilização de sistemas de software vem crescendo em complexidade. Em consonância a isso, sabe-se que sistemas de software são abstratos, intangíveis e podem se tornar extremamente complexos de forma muita rápida, difíceis de compreender e caros para se modificar [Sommerville 2011]. Tomando isso como motivação, a Engenharia de Software assume uma função essencial para garantir que tarefas, dados, pessoas e tecnologias associadas a todos os aspectos de produção do software estejam apropriadamente alinhados de forma efetiva e eficiente.

Os objetivos da Engenharia de Software se apoiam no desenvolvimento profissional de software, onde a seleção do método mais adequado para o conjunto de critérios envolvidos no processo de desenvolvimento deve ser analisado para ter sucesso com a abordagem adotada. Para isso, a Engenharia de Software propõe um conjunto de etapas que envolvem métodos, ferramentas e procedimentos, cujas estratégias de desenvolvimento, também conhecidas como paradigmas, auxiliam no desenvolvimento durante todo o ciclo de produção de um projeto. A escolha de um paradigma deve ter como base (i) a natureza do projeto e da aplicação, (ii) os controles e os produtos que precisam ser desenvolvidos e (iii) os métodos e ferramentas a serem utilizados. Dentre

os paradigmas, alguns são amplamente discutidos em literaturas e autores tradicionais, a saber: Ciclo de Vida Clássico ou Cascata, Prototipação, Modelo Espiral e Técnicas de Quarta Geração.

Retornando ao fato dos sistemas computacionais ficarem mais complexos diante das necessidades atuais, existe uma convergência do desenvolvimento de software para a utilização de ferramentas e métodos que auxiliem a criação sistemas inteligentes capazes de se adaptarem a diversos cenários hostis. Em meio a esse panorama, surge a Engenharia de Software Orientada a Agentes (ESOA) [Jennings 2000]. Essa área fornece soluções para o desenvolvimento de agentes inteligentes que operam em ambientes com constantes mudanças, aleatórios e abertos. Estes agentes devem ter a capacidade de planejar objetivos e decidir por si só quais ações utilizar para a obtenção dos objetivos.

Nas unidades posteriores serão detalhados alguns pontos acima citados, como engenharia de software orientada a agentes, desenvolvimento de agentes inteligentes, ambientes de agentes e sistemas multi-agentes. Além disso, serão trabalhados conteúdos relativos à estudos de casos para ilustrar o uso das ferramentas de desenvolvimento de agentes apresentadas.

2. Engenharia de Software Orientada a Agentes (ESOA)

A exigência de sistemas de software de qualidade e em escala industrial tem tornado sua construção cada vez mais difícil. Tal complexidade deixa a construção de software entre uma das atividades mais complexas que o ser humano é capaz de fazer. Visando diminuir essa dificuldade, a engenharia de software vem criando uma gama de paradigmas a serem utilizados em seu desenvolvimento (a saber, programação estruturada, programação declarativa, programação orientada a objetos, etc.). Cada um desses paradigmas representa a evolução do processo de desenvolvimento. No entanto, a busca por novas técnicas que possam tornar o desenvolvimento de software mais eficiente é uma atividade incessante para os engenheiros de software. E um dos paradigmas que mais vem sendo explorado pelos pesquisadores é o paradigma orientado a agentes.

Nesta unidade será analisado o paradigma orientado a agentes, englobando desde os conceitos básicos até as diferentes arquiteturas do agente, detalhando melhor a arquitetura BDI.

2.1 Conceitos e Fundamentos dos Agentes Inteligentes

Um agente é uma entidade capaz de perceber um ambiente através de sensores e atuar sobre o mesmo através de atuadores [Russel e Norvig 2009]. No entanto, para programa-lo é preciso entender como ele deve se comportar no seu ambiente e esse comportamento é descrito através da função do agente. É possível imaginar essa função como uma tabela contendo todas as possíveis percepções do agente e, a partir dela, deve-se mapear suas ações de acordo com essas percepções. Para ilustrar essa ideia toma-se como exemplo o agente aspirador de pó ilustrado na figura 1. O ambiente em que ele se encontra é muito simples, existem duas localidades: os quadrados A e B. O agente é capaz de perceber em que quadrado ele se encontra e se o mesmo está sujo. Ele pode realizar as seguintes ações, *(i)* limpar o quadrado caso ele esteja sujo, ou *(ii)* ir para o quadrado ao lado. Na figura 2 a função do agente é mapeada em uma tabela.

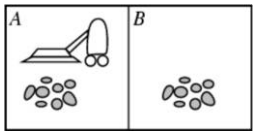


Figura 1. Agente aspirador de pó.

Sequência de Percepções	Ação
[A, Limpo]	Direita
[A, Sujo]	Aspirar
[B, Limpo]	Esquerda
[B, Sujo]	Aspirar
[A, Limpo], [A, Limpo]	Direita
[A, Limpo], [A, Sujo]	Aspirar
...	
[A, Limpo], [A, Limpo], [A, Limpo]	Direita
[A, Limpo], [A, Limpo], [A, Sujo]	Aspirar
...	

Figura 2. Tabela de funções do agente.

Mas o que torna um agente diferente de um objeto ou de um procedimento? Afinal de contas um agente é programado da mesma forma que qualquer outro software? Estas perguntas remetem a uma reflexão de que um objeto poderia cumprir as mesmas tarefas dadas a um agente, porém uma das diferenças entre agentes e não agentes está no ambiente. Agentes são capazes de se adaptar as mudanças provocadas pelo ambiente, realizando ações para que seus objetivos sofram o menor impacto possível. Essa característica do agente é chamada de adaptabilidade.

Um agente racional é um agente que faz a coisa certa, onde certo é qualquer meio que o leve a cumprir seu objetivo. Pode-se classificar o certo apenas como um agente agindo de acordo com sua função agente. Quando um agente esta inserido em um ambiente e recebe uma sequência de percepções, ele reage com uma sequência de ações que faz com que o ambiente passe por uma sequência de estados. Se esses estados forem desejáveis pode-se dizer que o agente teve um bom desempenho.

Esse desempenho do agente descreve o quão bem ele cumpriu seu objetivo. Não existe uma medida padrão de desempenho para um agente, cabendo ao desenvolvedor adequá-lo ao seu problema. A grande dificuldade é que nem sempre é fácil medir o desempenho do agente. Tomando o agente aspirador de pó como exemplo, uma medida de desempenho pode ser dada pelo número de vezes que ele limpou o chão em um determinado intervalo de tempo, mas, o agente poderia trapacear limpando o chão e em seguida jogando o lixo fora no mesmo local e limpando novamente. Uma medida adequada poderia recompensar o agente por deixar um quadrado limpo, talvez dando alguma penalidade por consumo de energia.

Para que um agente aja de forma racional, deve-se inserir nele as seguintes características:

- A medida de desempenho que define o seu critério de sucesso.
- O conhecimento prévio do ambiente.
- As ações que o agente pode utilizar.
- A sequência de percepções do agente.

Para cada sequência de percepção possível, um agente racional deve selecionar uma ação em que espera maximizar sua medida de desempenho, dada a evidência fornecida pela sequência de percepção e o seu conhecimento.

Todas as ações do agente dependem de suas percepções. E para que o agente não tome decisões inteligentes é necessário fazer com que o mesmo busque informações sobre o ambiente antes de realizar uma ação. Por exemplo, supõe-se que o objetivo de um agente seja atravessar uma rua. Caso ele não olhe para os lados antes de atravessá-la ele não perceberá um carro e poderá ser atropelado. A busca de informações garante que o agente tenha o máximo de percepções do ambiente possível e pode fazer com que o mesmo ganhe conhecimento tornando-o mais autônomo.

2.2 Arquiteturas de Agentes

Dessa forma, o comportamento de agentes e suas ações são processados após uma sequência de percepções, mas, como seria o funcionamento interno de um agente? Nesta unidade é discutido o funcionamento abstrato e a estrutura interna dos agentes segundo os modelos mais difundidos na literatura [Russel e Norvig 2009] [Wooldridge 2008].

Todos os agentes de uma maneira geral seguem a mesma macro arquitetura abordada no inicio da unidade, pois eles percebem o ambiente através de sensores e atuam sobre o mesmo através de atuadores. Um frente classifica esses agentes em quatro tipos: (i) agentes reativos, (ii) agentes baseados em modelo, (iii) agentes baseados em objetivos e (iv) agentes baseados em utilidade [Russel e Norvig 2009].

2.2.1 Agentes Reativos

Os agentes reativos respondem continuamente as mudanças do ambiente. Os agentes selecionam suas ações baseados na sua percepção, ignorando suas percepções anteriores. Agentes reativos tem a enorme vantagem de serem muito simples. Mas, sua inteligência é muito limitada. Como cada ação é baseada em uma percepção, se esta percepção não puder ser observada pelo agente, podem ocorrer sérios problemas durante sua execução. A figura 3 exemplifica a arquitetura de um agente reativo.

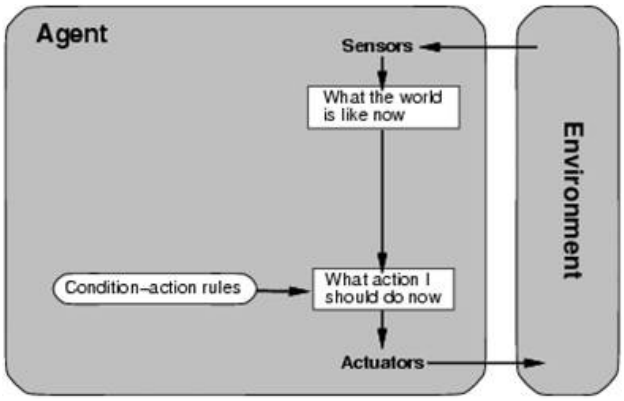


Figura 3. Arquitetura de um agente reativo.

2.2.2 Agentes Baseados em Modelos

Para solucionar o problema da falta de percepção do agente reativo, foi introduzida em sua arquitetura uma memória das percepções do agente. Assim, o agente pode manter um estado interno que depende do seu histórico de percepções. Assim ao tomar uma decisão o agente deverá levar em conta o seu estado interno e sua percepção atual do ambiente. Na figura 4 é ilustrada a arquitetura de um agente baseado em modelos.

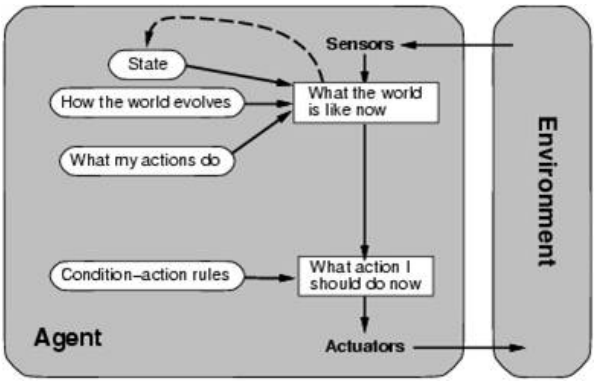


Figura 4. Arquitetura de um agente baseado em modelos.

2.2.3 Agentes Baseados em Objetivo

Em algumas situações conhecer o estado do ambiente não é o suficiente para que o agente decida que ação deve tomar, por exemplo, que rota tomar para chegar a um determinado local. Para o agente poder decidir qual rua ele deve seguir além de perceber que rua ele está, é exigido o conhecimento do lugar onde deve ir. Diferente dos agentes anteriores, o agente baseado em objetivo dever considerar o que pode acontecer no futuro. Ele deve poder estimar se suas ações o deixarão mais próximo do seu objetivo. Na figura 5 pode ser percebida a arquitetura de um agente baseado em objetivo.

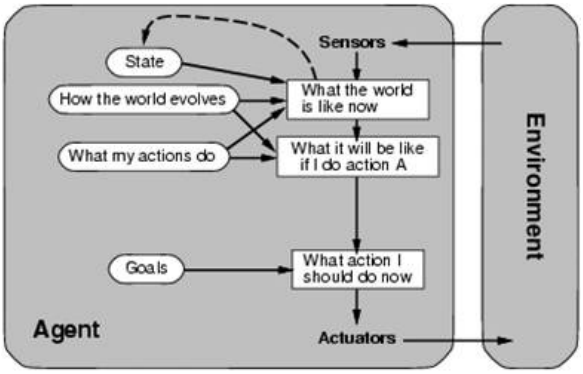


Figura 5. Arquitetura de um agente baseado em objetivo.

2.2.4 Agentes Baseados em Utilidade

A maioria dos ambientes para os agentes baseados em objetivos não são suficientes para gerar comportamentos de alta qualidade. Utilizando o exemplo anterior é possível usar várias rotas para chegar a um destino, mas existem rotas mais rápidas e outras mais seguras. Objetivos só podem ser descritos como realizados ou não e uma medida de performance poderia permitir saber o quão bom um objetivo foi atingido. Essa medida de performance é chamada de utilidade. A função utilidade mapeia os estados do agente em números reais que descrevem a qualidade do comportamento do agente. Essa função também pode ser utilizada em casos onde o agente possui objetivos conflitantes, como velocidade e segurança, realizando um balanceamento entre estes. Na figura 6 tem-se a ilustração da arquitetura de um agente baseado em utilidade.

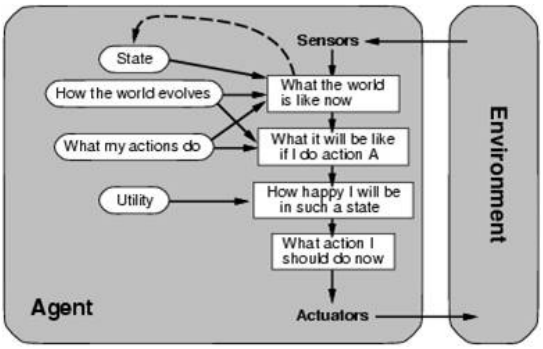


Figura 6. Arquitetura de um agente baseado em utilidade.

2.3 Arquitetura BDI

Uma outra arquitetura de agentes bastante difundida é a arquitetura BDI (*Belief-Desire-Intention*). Ela é baseada em um modelo cognitivo fundamentado em três atitudes mentais que são crenças, desejos e intenções (abreviadas por BDI, *belief, desire, intentions*) [Wooldridge 2008]. As crenças do agente representam o seu conhecimento em relação ao ambiente ou a si mesmo. Os desejos representam os objetivos que o agente deseja alcançar e as intenções representam a sequência de ações que o agente irá seguir para alcançar seus objetivos [Bordini e Viera 2003]. A figura 7 representa a arquitetura abstrata de um agente BDI.

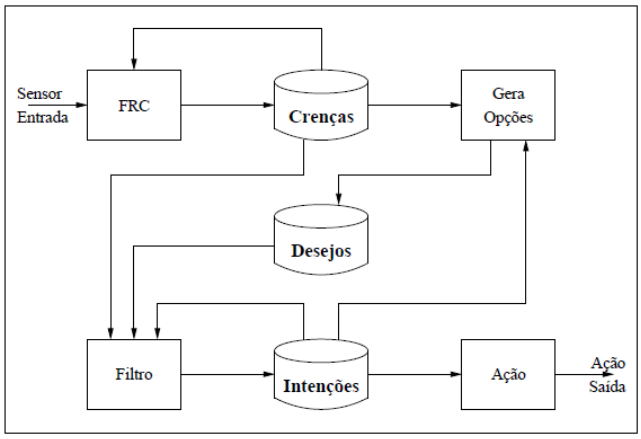


Figura 7. Arquitetura abstrata de um agente BDI.

A função de revisão de crenças (FRC) recebe informação sensorial do ambiente e consulta as crenças anteriores do agente (estados anteriores). Em seguida essas crenças são atualizadas para que reflitam o novo estado do ambiente. Com essa nova informação o agente verifica, através da função Gera Opções, quais serão seus objetivos (Desejos). Com os objetivos já definidos o agente pode definir as ações a serem tomadas para que os mesmos sejam cumpridos (Intenções). A função Filtro evita que o agente se comprometa com intenções incoerentes, intenções as quais ele já se comprometeu e que não podem ser realizadas. Com essas intenções filtradas a escolha entre as ações a serem tomadas é executada na função Ação.

Para casos simples essa escolha pode ser simples, bastando apenas o agente escolher uma das intenções ativas, desde que se garanta que em algum momento todas as intenções sejam executadas. Para casos mais complexos, onde existem desejos

conflitantes por exemplo, o agente deverá adotar um critério mais sofisticado para garantir que essas intenções sejam realizadas.

3. Plataforma de Desenvolvimento de Agentes

Esta unidade descreve a plataforma de desenvolvimento que foi utilizada para implementar os agentes com seus ambientes no decorrer deste documento. Basicamente, a plataforma é composta por uma linguagem de programação chamada **AgentSpeak**, inspirada e projetada para programação de agentes com a arquitetura BDI, especificamente, uma versão estendida da linguagem AgentSpeak, que fornece recursos para o desenvolvimento de sistemas multi-agentes, chamada **Jason**. A principal ideia da programação em AgentSpeak é definir o **conhecimento** do programa sob a forma de **planos**, ou seja, como esse programa resolverá determinados problemas para atingir seus objetivos. Planos em AgentSpeak também são usados na caracterização de respostas para eventos e, desta forma, seguir instruções que sistematicamente tentam atingir os objetivos utilizando o conhecimento descrito previamente.

O paradigma do AgentSpeak incorpora aspectos convencionais da programação imperativa, consagrada na linguagem C, bem como aspectos da programação declarativa (estilo dedutivo), tal como na linguagem Prolog. No entanto, o estilo de programação em AgentSpeak é essencialmente diferente destes dois paradigmas. Esta diferença pode ser resumida da seguinte forma: programas em AgentSpeak são **agentes**. Isso porque são peças ativas de software que estão tentando constantemente atingir os objetivos em seu ambiente fazendo uso de planos ou subplanos.

Abaixo alguns dos requisitos que o AgentSpeak e Jason se propõem a cumprir, mencionados por [Bordini, Hübner e Wooldridge 2007]:

- Suportam delegação em nível de objetivos. Quando uma tarefa é delegada a um agente, não se faz isso fornecendo ao agente uma descrição executável do que realizar, como no paradigma imperativo. Ao invés disso, a comunicação com ele é realizada em nível de objetivos, pois deseja-se que os agentes sejam capazes de agir autonomamente de forma a alcançar os objetivos delegados. Para isso, se faz necessário descrever os objetivos em alto nível, independente das abordagens para atingi-los;
- Produzem sistemas que são sensíveis ao seu ambiente de execução;
- Sintaxe limpa, direcionada à objetivos e possui comportamento responsivo;
- Suportam comunicação em nível de conhecimento e cooperação.

Estas são as principais características que o AgentSpeak e o Jason fornecem para dar suporte à programação de agentes autônomos. Nas próximas unidades é fornecida uma breve introdução para exemplificar tais requisitos.

3.1. A Linguagem AgentSpeak

A linguagem de programação **AgentSpeak** foi proposta inicialmente por [Rao 1996]. A linguagem é uma extensão natural e elegante de programação em lógica para arquitetura de agentes BDI na forma de **sistemas de planejamento reativo**. Sistemas de planejamento reativo são sistemas que estão sempre em execução, reagindo a eventos que acontecem no ambiente através da execução de planos. Um agente em AgentSpeak possui uma **base de crenças** inicial e um **conjunto de planos**. Um **átomo de crença** é

um predicado de primeira ordem na notação lógica usual, e **literais de crença** são átomos de crenças ou suas negações. Portanto, a base de crenças de um agente é uma coleção de átomos de crença [Bordini e Viera 2003].

Na programação orientada à agentes, a noção de **objetivos** é fundamental. Normalmente ao se representar um objetivo '*x*' em um programa agente, significa que o agente deve agir de modo a mudar o mundo para um estado que, ao perceber o ambiente, ele acredite que '*x*' é verdade [Bordini, Hübner e Wooldridge 2007]. Existem dois tipos de objetivos em AgentSpeak: **objetivos de realização** e **objetivos de teste**, ambos são predicados, assim como as crenças, porém com operadores prefixados '**!**' e '**?**', respectivamente. Na prática, esses objetivos iniciam a execução de **subplanos**. Objetivos de realização expressam que o agente quer atingir um estado no ambiente em que o predicado associado ao objetivo é verdadeiro. Já um objetivo de teste retorna a unificação do predicado de teste com uma crença do agente, ou falha caso não seja possível. Crenças e objetivos são duas importantes **atitudes mentais** que podem ser expressas no código fonte de um agente. A terceira construção essencial de um programa agente, no estilo BDI, são os planos ("know-how" do agente).

Um plano em AgentSpeak são ações básicas que um agente é capaz de executar em seu ambiente. É constituído por três partes distintas: um **evento ativador** (propósito do plano), um **contexto** (conjunção de literais de crenças), e um **corpo** (sequência de ações básicas ou subobjetivos). As três partes do plano são separados por ':' e '<-' e finalizado com um ponto, como segue: " *evento_ativador* **:** *contexto* **<-** *corpo* .". Um **evento ativador** define quais eventos podem iniciar a execução de um plano. Um evento pode ser **interno**, quando gerado pela execução de um plano em que um subobjetivo precisa ser alcançado, ou **externo**, quando gerado pelas atualizações de crenças que resultam das percepções do ambiente. Eventos representam mudanças de crenças e objetivos. Essas mudanças podem ser de adição ou remoção de atitudes mentais representadas pelos operadores prefixados '+' e '-', respectivamente.

Um exemplo clássico pode ser visualizado em um programa que calcula o fatorial de um número inteiro. O fatorial é um exemplo bem ilustrativo pois ele demonstra o uso da recursão, característica marcante de algumas linguagens de programação. Uma versão eficiente para computar o fatorial é ilustrada por [Bordini, Hübner e Wooldridge 2007] e mostrada logo abaixo na figura 8.

```
!print_fact(5).

+!print_fact(N)
   <- !fact(N,F);
      .print("Factorial of ", N, " is ", F).

+!fact(N,1) : N == 0.

+!fact(N,F) : N > 0
   <- !fact(N-1,F1);
      F = F1 * N.
```

Figura 8. Código AgentSpeak para computar fatorial.

A primeira linha define o objetivo inicial do agente (o ponto de exclamação indica que este é um objetivo a ser alcançado). Portanto, o objetivo inicial do agente é

imprimir o fatorial de 5. Como subplano, calcula-se o fatorial de N e, em seguida, imprime o resultado, como indicado no primeiro plano.

3.2. Jason

Jason é uma linguagem de programação baseada em JAVA que implementa e estende a linguagem AgentSpeak para prover uma plataforma de desenvolvimento de sistemas multi-agentes, incluindo a integração com diversas tecnologias, como JADE e Moise+. Em Jason, as funções de seleção de eventos, aplicação de planos e intenções, pertencentes ao ciclo de deliberação dos agentes, são customizáveis e podem ser implementadas como métodos JAVA. Além disso, Jason fornece uma estrutura para implementação de manipuladores de percepções e de eventos externos também na forma de métodos JAVA.

Um agente opera por meio de um ciclo de raciocínio e, no caso do Jason, é dividido seu ciclo em de dez passos principais nos quais o interpretador executa um programa agente [Bordini, Hübner e Wooldridge 2007]. As dez etapas do ciclo de raciocínio são:

1. *Percepção* do ambiente;
2. *Atualização* da base de crenças;
3. *Comunicação* com outros agentes;
4. *Seleção* de mensagens "*socialmente*" aceitáveis;
5. *Seleção* de um evento;
6. *Recuperação* de todos os planos relevantes;
7. *Determinação* dos planos aplicáveis;
8. *Seleção de um* dos planos aplicáveis;
9. *Seleção de uma* intenção para execução futura;
10. *Execução de um passo* da intenção;

O **passos de 1 a 4** se referem à preocupação dos agentes em atualizar suas crenças à respeito do seu ambiente e dos outros agentes envolvidos. Já os **passos de 5 a 10**, descrevem as principais etapas da interpretação de programas AgentSpeak. O processo começa com a base de crenças e o conjunto de eventos sendo atualizados e, em seguida, um dos eventos é selecionado para ser tratado (levando a uma intenção nova ou atualizada), e uma das intenções do agente é executada.

A primeira operação que um agente realiza dentro do ciclo de raciocínio é perceber o seu ambiente, atualizando suas crenças sobre o estado em que o ambiente se encontra. Essas percepções ocorrem sob a forma simbólica, através de uma lista de literais. Cada literal representa um percepção, que é uma representação simbólica de uma propriedade particular do estado atual do ambiente. Uma vez que a lista de percepções foi obtida, a base de crenças precisa ser atualizada para que possa refletir as mudanças percebidas no ambiente. Cada alteração na base de crença gera um **evento**. Em particular, o tipo de evento gerado em virtude da adição ou exclusão de crenças durante as atualizações da base de crenças decorrentes da percepção do ambiente, é chamado de **evento externo** (eventos internos estão associados à intenções).

Outras importantes fontes de informações para um agente em um sistema multiagente são outros agentes que participam do mesmo sistema. Nesta fase do ciclo de raciocínio, o interpretador Jason verifica que mensagens devem ser entregues para outros agentes. Isso é feito pelo método *checkMail* que simplesmente tornam as mensagem recebidas disponíveis a nível do interpretador AgentSpeak. Por padrão, o Jason irá processar as mensagens de comunicação na ordem em que foram recebidas. Antes das mensagens serem processadas, elas passam por uma seleção para determinar se podem ser aceitas ou não pelo agente. A função de seleção das mensagem, por padrão, aceita todas as mensagens de todos os agentes, no entanto, ela pode ser personalizada.

Agentes BDI operam continuamente através da manipulação de eventos. Eventos representam mudanças percebidas tanto no ambiente quanto nos objetivos dos agentes e podem acontecer vários eventos pendentes por conta de mudanças em diferentes aspectos do ambiente, contudo, apenas um evento pendente pode ser tratado por ciclo de raciocínio. O conjunto de eventos é implementado em Jason como uma lista, e novos eventos são adicionados ao fim dessa lista. Por padrão, a função de seleção de eventos sempre escolhe o primeiro evento da lista para tratar, no entanto, a função pode ser personalizada pelo programador. No momento em que tem-se um evento selecionado é preciso encontrar um plano que irá permitir que o agente atue de forma a lidar com esse evento. O próximo passo a ser feito é encontrar na **biblioteca de planos**, todos os planos que são relevantes para o dado evento e isto é realizado por meio da obtenção de todos os planos da biblioteca de planos do agente que possui um **evento de disparo** e pode ser unificado com o evento selecionado.

Assim, apesar de ter selecionado todos os planos, não é possível utilizar qualquer um desses planos para ser a ação que o agente tomará para lidar com o evento selecionado. É necessário ainda selecionar os planos atualmente aplicáveis, ou seja, um plano que, dado o conhecimento do agente e suas crenças atuais, parece ter uma chance de sucesso. Para isso é preciso verificar se o contexto é consequência lógica da base de crenças do agente. Portanto, o agente precisa escolher um dos planos aplicáveis e comprometer-se a executá-lo. Em Jason, intenções são pilhas de planos que representam um curso parcial de ações. Normalmente um agente tem mais de uma intenção no conjunto de intenções, cada uma representando um foco diferente. No entanto, somente uma intenção deve ser executada por ciclo de raciocínio, logo, deve-se escolher uma intenção específica entre aquelas atualmente prontas para execução. Assim, a implementação padrão do Jason para seleção de intenções escolhe cada uma das intenções por vez para executar.

Por fim, a execução de uma intenção específica depende do tipo de fórmula que existe no início do corpo do plano presente na intenção selecionada. O que o interpretador fará dependerá de cada tipo de fórmula. Existem seis tipos que podem aparecer no corpo de um plano, são elas: *ações ambiente, objetivos de realização, objetivos de teste, notas mentais, ações internas e expressões*. Antes que outro ciclo de raciocínio inicie, o interpretador verifica se qualquer feedback está disponível e, nesse caso, as intenções relevantes são atualizadas e incluídas novamente ao conjunto de intenções para que tenham chance de serem selecionadas para execução no ciclo de raciocínio seguinte.

4. Ambientes de Agentes

Como descrito anteriormente, os agentes necessitam habitar ou visitar um ou vários ambientes de forma a perceber a ocorrências de mudanças e, com base nelas, atuar significativamente na realização de alguma(s) tarefa(s). Um ambiente em que um agente esteja inserido pode ser classificado sob alguns aspectos que irão determinar o seu tipo e definir o quão complexo pode ser esse ambiente [Russel e Norvig 2009]. No entanto, em um ambiente simulado, baseado em agentes, o número de aspectos a ser considerado deve ser suficientemente grande a fim de promover confiabilidade para a obtenção dos resultados e das conclusões. Um exemplo disso são os jogos de computador. Neles, a indústria já entendeu que a natureza do ambiente é essencial, assim como a qualidade gráfica, os detalhes dos cenários, a organização do espaço ou a sonorização [Coelho 2008].

As raízes teóricas dos ambientes os definem como sendo um problema e os agentes que atuam sobre ele como sendo a provável solução. Durante a concepção de um agente, o ambiente de tarefas deve ser o mais bem especificado possível, detalhando elementos como: medida de desempenho, ambiente, atuadores, sensores, objetivos, ações e percepções [Russel e Norvig 2009]. As propriedades determinantes para caracterizar um ambiente estão associadas conforme os itens abaixo:

- **Quanto à sua observabilidade**: um agente pode perceber o ambiente usando seus sensores e essa percepção pode ser total, parcial ou nenhuma.
- **Quanto à atuação de um ou mais agentes**: o número de agentes atuando em um determinado ambiente o classifica em mono ou multi-agente. O comportamento dos agentes em um ambiente multi-agente pode ser do tipo cooperativo ou competitivo, onde no primeiro os agentes se ajudam em busca do cumprimento de um objetivo comum. No segundo, os agentes competem entre si em busca de uma melhor solução, medida por uma função de avaliação.
- **Quanto à mudança de estado do ambiente:** se o próximo estado do ambiente é completamente determinado pelo estado atual somado com as ações do agente então este é um ambiente determinístico. Caso contrário é um ambiente estocástico.
- **Quanto às observações durante as mudanças de estado do ambiente**: em um ambiente de tarefa episódico, a experiência do agente é divida em episódios atômicos onde em cada episódio que o agente recebe uma percepção é então selecionada uma ação. Desse modo, o próximo episódio não depende de ações executadas em outros episódios anteriores. Caso isso não seja verdade então o ambiente é sequencial.
- **Sobre as mudanças no próprio ambiente**: se o ambiente pode ser modificado enquanto o(s) agente(s) delibera(m) então ele é um ambiente dinâmico. Caso contrário é um ambiente estático.
- **Sobre o número de estados possíveis de um ambiente**: está relacionado com o número de estados possíveis, o modo como o tempo é tratado e ainda as percepções e ações do(s) agente(s). Se o número de possíveis estados for finito então este é um ambiente discreto, caso contrário é um ambiente contínuo.

- **Quanto ao conhecimento do agente em relação ao ambiente**: está relacionado em como o agente conhece as leis do ambiente. Em um ambiente conhecido os resultados das ações de todos os agentes são igualmente conhecidas. Obviamente, se o ambiente é desconhecido então o agente terá que aprender como ele funciona para realizar ações relevantes.

Assim, nota-se que os ambientes de agentes podem ser compreendidos tanto como computacional quanto físico. Esta unidade trata especificamente dos ambientes de agentes sob a forma computacional, podendo ser materializado em bases de dados, conjuntos de arquivos, páginas da internet, etc.

4.1. Ambientes sob a perspectiva da Engenharia Orientada a Agente

Atualmente na literatura sobre agentes inteligentes existem basicamente duas abordagens para a definição de ambientes de agentes. Uma oriunda das raízes clássicas da Inteligência Artificial e outra, mais atual, vinda da área especificada como Engenharia de Software Orientada a Agentes (AOSE). A primeira define as noções de ambientes como um mundo externo que é percebido e modificado por agentes de forma a cumprir com seus objetivos ou tarefas. Na segunda é introduzido uma noção de ambiente como uma **abstração de primeira classe** para a engenharia de sistemas multi-agentes (SMA). Isso implica na existência de um lugar adequado para encapsular funcionalidades e serviços que suportam atividades de agentes [Ricci 2011].

De fato um ambiente provê as condições necessárias para que uma entidade exista, seja ela um Agente ou um Objeto. Ele define e fornece as propriedades do mundo no qual o agente irá atuar. Sem um ambiente o agente é totalmente inútil, pois não haverá sensações nem ações. Com isso, projetar um agente eficaz engloba uma análise holística dos aspectos físicos e computacionais do ambiente, pois um ambiente de agente não consiste em apenas um conjunto de outras entidades, mas também nos princípios e processos que definem a própria existência dos agentes, assim como, a forma como eles se comunicam [Odell 2003].

Em AOSE o ambiente não é apenas um alvo das ações do agente, nem um *container* do tipo gerador de percepções para os agentes, mas sim parte do SMA. Este ponto de vista define o ambiente como **endógeno** em relação ao SMA, pois o mesmo faz parte do sistema para o qual foi projetado.

4.2. Ambientes baseados em artefatos

Sob a perspectiva da AOSE, o lugar adequado para encapsular funcionalidades e serviços a serem explorados por agentes é constituído dinamicamente por entidades computacionais, denominadas **Artafetos**. O conjunto de artefatos que formam o ambiente do agente, pode ser organizado em um ou várias áreas de trabalho (***workspace***) distribuídas por uma infraestrutura local ou remota.

Os artefatos estão caracterizados fundamentalmente em termos de operações, propriedades observáveis, sinais e manual (*operations*, *observable property*, *observable event* e *manual*, respectivamente). Programadores de SMA definem os tipos de artefatos de forma bastante análoga à Programação Orientada à Objetos, pois se faz necessário estabelecer elementos estruturais e comportamentais para cada artefato.

Dessa forma, para possuir as funcionalidades disponíveis e exploráveis pelos agentes, os principais elementos de um artefato são:

- ***Operations***: representam as operações processuais executadas dentro de um artefato. Podem ser do tipo interna (apenas o próprio artefato pode dispará-la) ou externa (disparada por um agente ou outros artefatos);
- ***Observable property***: uma propriedade observável está associada com as variáveis de estado do artefato. Seu valor pode ser percebido pelos agentes que monitoram o artefato.
- ***Observable event***: a execução de uma operação pode gerar um ou vários sinais. Estes sinais são eventos observáveis não persistentes e podem ser percebidos pelos agentes.
- ***Manual***: um artefato pode ser equipado por um manual. Uma espécie de documento legível por máquina (os agentes) contendo uma descrição das suas funcionalidades, assim como, a forma de como usá-las.

Com isso, os agentes podem atuar em *workspaces* e a realizar ações em um ou vários artefatos de ambientes. Existem basicamente três grupos de ações possíveis, são elas: (1) criar, pesquisar e eliminar artefatos; (2) usar, executar operações e observar propriedades e/ou eventos; (3) vincular ou desvincular artefatos entre si.

4.3. Ambientes CArtAgO

Este modelo de programação de ambientes baseado em artefatos pode ser implementado no **Cartago** (*Common ARtifact infrastructure for AGent Open environments*). Cartago é um framework e infraestrutura para programação e execução de ambientes desenvolvido conforme o modelo informal descrito anteriormente. É um *framework* pois oferece uma Interface de Programação de Aplicativos (API) em Java para programar artefatos e ambientes em tempo de execução e executar ambientes baseados em artefatos, além de uma biblioteca com um conjunto de tipos de artefatos de uso geral pré-definidos. É também uma infraestrutura pois provê uma API e um mecanismo subjacente para estender linguagens/*frameworks* de programação de agentes.

A programação de um Artefato é feita a partir da extensão da classe **cartago.Artifact** usando um conjunto básico de anotações Java e métodos herdados para definir os elementos estruturais de um artefato e seus comportamentos. O uso de anotações Java facilita o processo indutivo para a programação dos recursos como, por exemplo, para que um método seja reconhecido pelos agentes como uma operação do ambiente ele recebe em sua assinatura a anotação @OPERATION.

Os agentes Jason possuem total integração com as funcionalidades do Cartago por meio do mapeamento das propriedades observáveis, sinais e operações. A troca de dados entre ambos é feita de forma natural para alguns tipos definidos, como por exemplo, booleans são mapeados entre eles como booleans, arrays são mapeados como listas e objetos em geral são mapeados como átomos para o Jason. Um agente Jason que necessite "habitar" um artefato de ambiente Cartago deve procurá-lo executando uma ação do tipo **lookupArtifact (<nome_do_artefato>, <identificador_local_para_o_agente>)**. Feito isso, o agente deve efetivamente "entrar" no artefato usando o comando **focus(<identificador_local_para_o_agente>).**

5. Estudo de caso com Jason e CArtAgO

Esta unidade objetiva ilustrar o uso e a integração das tecnologias apresentadas nas últimas duas unidades por meio da construção de dois estudos de casos específicos.

O primeiro estudo de caso envolve um agente em um ambiente e possui os seguintes aspectos técnicos: (a) uso da internet como ambiente do agente; (b) uma ação interna do agente que faz uso de uma biblioteca auxiliar de uma rede social (Twitter). O segundo estudo engloba dois agentes em uma aplicação que simula um jogo de "bingo". Ambos os agentes possuem suas crenças bem definidas, realizam comunicações e agem conforme suas percepções do ambiente.

5.1. Estudo de caso I: Um agente monitor de cardápio para um restaurante universitário

O ambiente deste estudo é do tipo computacional. Possui as propriedades parcialmente observável, episódico, estocástico, contínuo e contém os dados oriundos de uma página HTML disponível na internet sob o endereço: http://www.uece.br/uece/index.php/restauranteuniversitario. Neste endereço é possível encontrar o cardápio da semana para a comida servida no Restaurante Universitário da Universidade Estadual do Ceará, Campus Itaperi. Na maioria do tempo esse ambiente é estático e por isso o agente realiza consultas em um período longo de tempo.

O objetivo inicial do agente batizado de Linguini é monitorar um artefato de ambiente a fim de perceber os dados relacionados com o cardápio da semana. Inicialmente, a sua base de crença estará vazia e por isso tudo que ele perceber em relação ao cardápio será absorvido como crença para posterior divulgação por meio de sua rede social. O monitoramento será realizado em intervalos de 24 horas e a cada nova percepção o agente fará uma nova publicação em sua rede social. Os detalhes do ambiente e a integração com o agente Linguini podem ser visualizadas na figura 9.

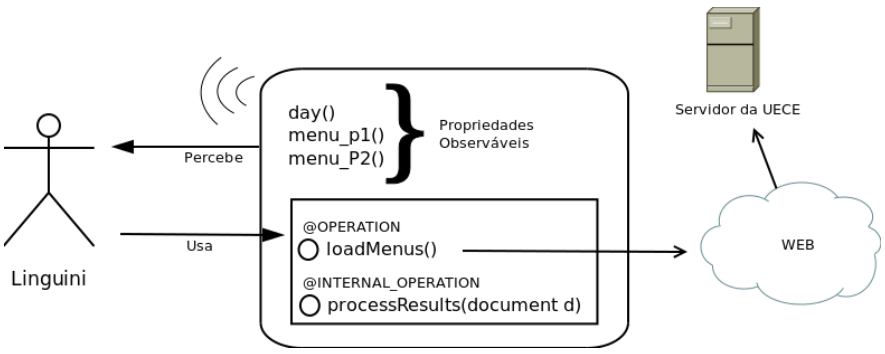


Figure 9. Detalhes do artefato de ambiente e a integração com o agente Linguini para o estudo de caso 1.

O artefato possui: três propriedades observáveis; uma operação que pode ser executada por um agente; uma operação interna do ambiente. As propriedades perceptíveis pelo agente se referem ao dia do cardápio e o conteúdo do menu dividido em duas partes. A operação utilizável por agentes realiza a carga do menu disponível na Internet por meio do protocolo HTTP e a operação interna efetua o processamento das respostas com o objetivo de filtrar os conteúdos vindos do servidor da UECE. Um recorte da codificação do Artefato (Cartago) para os elementos especificados podem ser visualizadas na figura 10, especificamente: (1) o método ***init*** que define as propriedades observáveis do ambiente; (2) o método ***loadMenus*** que encapsula o processo de comunicação com o servidor da UECE; (3) o método ***processResults*** que efetiva a filtragem dos resultados vindos da internet e atualiza os valores das propriedades observáveis (linhas 66, 69, 70, 73 e 74) para posterior percepção do agente.

```java
package workspaces;

import java.io.BufferedReader;

public class EnvironmentWeb extends Artifact { (1)
  void init() {
    defineObsProperty("day", "");
    defineObsProperty("menu_p1", "");
    defineObsProperty("menu_p2", "");
  }

  void loadMenus(){ (2)

  @INTERNAL_OPERATION
  void processResults(Document d){ (3)
    for (Element e : d.getElementsByClass("ru_head_title")){
      getObsProperty("day").updateValue(e.children().get(0).text());

      if (e.nextElementSibling().children().get(0).text().length() > 139){
        getObsProperty("menu_p2").updateValue(e.nextElementSibling().children().get(0).text().substring(139));
        getObsProperty("menu_p1").updateValue(e.nextElementSibling().children().get(0).text().substring(0, 139));
      }
      else{
        getObsProperty("menu_p1").updateValue(e.nextElementSibling().children().get(0).text().substring(0));
        getObsProperty("menu_p2").updateValue("");
      }

      await_time(10);
    }
  }
}
```

Figure 10. Trecho da codificação do artefato de ambiente.

Finalmente, a figura 11 descreve a codificação do agente na linguagem AgentSpeak do framework Jason. O segmento (a) define a base de crença inicial como literais de crenças vazios, (b) como as bibliotecas de planos iniciais e (c) a percepção da primeira parte do menu. No contexto da percepção é definido que o agente usará os valores percebidos para dia e a segunda parte do menu. Ao perceber o cardápio do dia, o agente efetua a divulgação dos dados de sua percepção em sua rede social executando a ação interna ***internalActions.sendTwitter.*** O plano de monitoramento entra em ação depois que o agente cria o artefato e carrega os cardápios iniciais. Nota-se nesta ação o ***looping*** do agente e sua espera por um novo dia.

```
/* Initial beliefs and rules */
day("").
menu_p1("").
menu_p2("").                                                    (a)

/* Initial goals */
!create.

/* Plans */
+!create: true <-
  ?setupArtifact(ID).                                           (b)

+?setupArtifact(E) : true <-
  makeArtifact("env_web", "workspaces.EnvironmentWeb", [], E);
  focus(E);
  !notification.

-?setupArtifact(E) : true <-
  .wait(30);
  !create.

+!notification: true <-
  loadMenus;
  !monitoring.

+!monitoring: true <-
  println("Esperando: ",(1000 * 60)/60000, " minutos");
  .wait(((1000 * 60) * 60 ) * 24);  //24 horas segundos
  !notification.
```

```
//Perceptions                                                   (c)
+menu_p1(MenuP1): day(Day) & menu_p2(MenuP2)<-
  if (Day \== ""){
    internalActions.sendTwitter(Day);
    internalActions.sendTwitter(MenuP1);
    if (MenuP2 \== ""){
      internalActions.sendTwitter(MenuP2);
    };
  }.
```

Figure 11. Código AgentSpeak do agente Linguini.

Objetivou-se neste primeiro estudo um exemplo simplório de demonstração do uso das ferramentas de desenvolvimento de agentes Jason com ambientes baseados em artefatos Cartago. Nele, foram codificados recursos de comunicação de ambiente com a Internet, integração agente-ambiente e comportamento interno de um agente usando a própria linguagem Java. Toda a codificação pode ser consultada em http://www.github.com/necioveras/linguini.

5.2. Estudo de caso II: Um simulador de um jogo de bingo

O ambiente deste estudo possui as propriedades parcialmente observável, sequencial, estocástico, contínuo e dinâmico, pois o próprio ambiente pode mudar enquanto o agente efetua as suas deliberações. O ambiente representa uma mesa de um jogo de "bingo" do mundo real. Ele contém propriedades observáveis por agentes, operações internas e operações exequíveis por agentes. Este ambiente também é capaz de emitir sinais (mensagens) aos agentes para comunicar acontecimentos importantes e de interesse geral, tais como, o sorteio de um número ou o status da mesa ("em sorteio", "parada", "vendendo cartelas" etc). Todo o processo de ***looping*** para sortear os números é realizado e controlado pelo próprio ambiente (dinamicidade) após uma autorização de um agente proprietário do bingo. A figura 12 permite obter uma visualização informal do funcionamento da arquitetura do estudo, incluindo o ambiente, agentes e suas integrações.

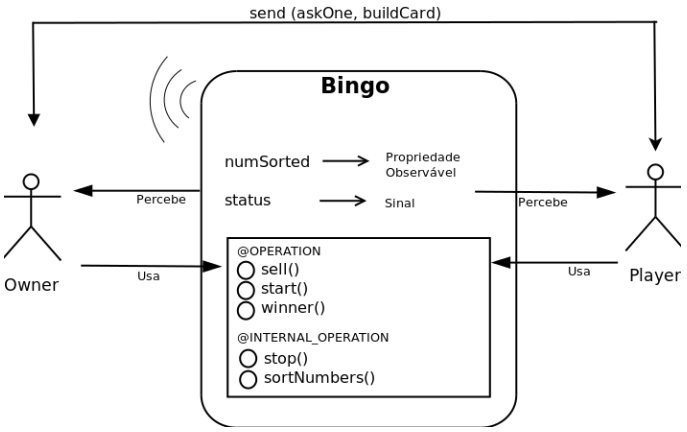


Figure 12. Arquitetura do estudo de caso 2.

O ambiente é criado pelo agente Owner, proprietário do Bingo. Ao criar, o agente monitora o ambiente à espera de um sinal *ready* para executar a operação *start* do ambiente e autorizar o início do sorteio. Este sinal é emitido quando o ambiente detecta que foram vendidas (operação *sell*) um número mínimo de cartelas para agentes Player's. Estes agentes por sua vez ficam "vagando" em busca de um artefato do tipo Bingo e, ao encontrar, solicitam uma cartela ao agente Owner por meio de uma mensagem (*send*). De posse da cartela, Player a adiciona em sua base de crença e realiza uma operação no ambiente registrando a sua aquisição (operação *sell*). Feito isso, Player apenas aguardará as mudanças do ambiente monitorando os sinais e as propriedades observáveis disponíveis.

A cada percepção de Player é gerada uma ação de comentários pessoais durante o sorteio, por exemplo, se ele acerta um número comenta que acertou e informa quantos números acertou no total. No final, quando algum jogador perceber que ganhou, este

mesmo jogador realiza a operação *winner* no ambiente para comunicar que ele é o vencedor.

Na figura 13 é mostrado a codificação do artefato de ambiente Bingo e em seguida, na figura 14, a codificação dos agentes Owner e Player. Para Onwer é encapsulado os planos de criação e teste de criação do artefato, pois esse artifício já foi demonstrado no estudo anterior. Nele é detalhado apenas a percepção do sinal *status* e o recebimento de uma mensagem do tipo *askOne* vinda do agente Player. No plano de recebimento da mensagem é executado a operação interna *ias.buildCard* e em seguida enviada uma mensagem de resposta para o agente solicitante da cartela informando os valores aleatórios gerados no presente cartão. Para o agente Player a figura mostra apenas os planos iniciais do agente. Suas ações, ativadas a partir das percepções, podem ser visualizada na figura 15.

```
package tablet;
import java.util.HashSet;
public class Bingo extends Artifact {

  private Set <Integer> box = new HashSet <Integer>();

  String internalStatus = "void";
  int MAXNumberSorted = 40;
  int MAXSold = 02;
  int sold = 0;

  void init() {
    defineObsProperty("numSorted", 0);
    signal("status", "selling");
    internalStatus = "selling";
  }

  @OPERATION
  void sell(){
    sold++;
    if (sold >= MAXSold){
      signal("status", "ready");
    }
  }

  @OPERATION
  void start(){
    signal("status", "started");
    internalStatus = "started";
    execInternalOp("sortNumbers");
  }

  @INTERNAL_OPERATION
  void stop(){
    internalStatus = "stop";
    signal("status", "stoped");
  }

  @OPERATION
  void winner(){
    internalStatus = "stop";
    signal("status", "stoped");
  }

  @INTERNAL_OPERATION
  void sortNumbers(){
    Random r = new Random();
    int counter = 0;
    while (!internalStatus.equals("stop")){
      int x = r.nextInt(100);
      if (box.add(x)){
        counter++;
        getObsProperty("numSorted").updateValue(x);
        signal("status", "sorted");
        await_time(3000);
      }
      if (counter >= MAXNumberSorted)
        execInternalOp("stop");
    }
  }
}
```

Figure 13. Codificação do artefato de ambiente Bingo.

```
/* Initial beliefs and rules */

sizeCard(math.random(9)+1).
maxNumberToSort(math.random(99)+1).

/* Initial goals */

!create.

/* Plans */

+!create : true <-

+?setupBingo (C) : true <-

-?setuBingo(C) : true <-

//Signal status
+status(S) :  S == "ready" <-
  start.

+!kqml_received(Sender, askOne, buildCard, Response) :
  sizeCard(Size) & maxNumberToSort(Max)
  <-
  ias.buildCard(Size, Max, Card);
  .send(Sender, tell, Card, Response).
```

```
/* Initial goals */

!participate.

/* Plans */

+!participate: true<-
  ?myArtifact (ID);
  focus(ID);
  .send(owner,askOne, buildCard, Card);
  +myCard(Card);
  sell;  //buy   - informa que adquiriu uma cartela
  println("Estou no bingo! Registrei minha cartela.").

+?myArtifact(C) : true <-
  lookupArtifact("b0", C).

-?myArtifact(Art) : true <-
  .wait(1000);
  println("Esperando por um bingo.");
  !participate.
```

Figure 14. Codificação dos agentes Owner (à esquerda) e Player (à direita).

```
23 //Perceptions of the signals
24
25 +myCard(Card) <-
26    .concat("Cartela:", Card,S);
27    println(S);
28    Hit = 0;
29    +myHits(Hit).                    //adiciona uma nova crença com o total de acertos
30
31 +status(S) : S == "sorted" & myCard(Card) & myHits(Hit) <-
32    ?numSorted(V);
33    println("Opa, percebi um numero sorteado ... ", V);
34    if (.member(V, Card) ) {
35       +myHits(Hit+1);
36       println("acertei:", V, " Ate agora acertei ", Hit+1, " numero(s) em um total de ", .length(Card));
37    }.
38
39 +myHits(Hit) : myCard(Card) <-
40    if (Hit == .length(Card)){
41       print("Gaaaaaaaaaaaaaaaaaaannnnnnnnnnnnhhhhhhhhhhheeeeeeeeiiiiiii!!");
42       winner;
43    }.
44
45
46 +status(S) :  S == "started" <-
47    println("Legal! Bingo inciado!").
48
49 +status(S) :  S == "stoped" <-
50    println("Ahhhh .. já acabou.").
51
52 //Percepctions of the Observable Properties
53 +numSorted(V).
```

Figure 15. Trecho de código de percepções do agente Player.

Visou-se neste estudo complementar um segundo exemplo de demonstração do uso das ferramentas de desenvolvimento de agentes Jason com ambientes baseados em artefatos Cartago. Nele, foram codificados dois agentes, recursos de comunicação entre agentes, integração agente-ambiente, comportamento interno de um agente e a definição de crenças como predicados de primeira ordem. Toda a codificação pode ser consultada em http://www.github.com/necioveras/Bingo_MAS_JaCa.

6. Introdução aos Sistemas Multi-Agentes

Nesta unidade são discutidos alguns conceitos iniciais relacionados aos Sistemas Multi-Agentes (SMA's) tais como, o conceito de Sociedade e Organizações em SMA's, a presença das normas no contexto organizacional e o modelo organizacional MOISE+.

6.1.Sociedades e Organizações

Sistemas Multi-Agente (SMA) são sistemas compostos de agentes inteligentes autônomos que possuem a capacidade de interagir com outros agentes presentes no sistema afim de desempenhar suas tarefas em um ambiente específico, podendo ser reativos ou proativos [Jennings e Wooldridg 1998]. Estes agentes exibem duas características fundamentais: (i) são capazes de agir de forma autônoma tomando decisões em prol de seus objetivos; (ii) são capazes de interagir com outros agentes utilizando protocolos de interação social inspirados nos humanos, os quais incluem pelo menos algumas das seguintes funcionalidades: coordenação, cooperação, competição e negociação.

Pode-se dizer ainda que um Sistema Multi-Agente é um sistema computacional em que dois ou mais agentes interagem ou trabalham em conjunto de forma a desempenhar determinadas tarefas ou satisfazer um conjunto de objetivos. A pesquisa científica e a implementação prática de Sistemas Multi-Agente está centrada na construção de métodos formais, princípios e modelos que permitam a representação de

pequenas e grandes sociedades de agentes autônomos, capazes de interagir convenientemente de forma a atingirem os seus objetivos [Lesser 1999].

Um ponto significativo para a construção de sociedades de agentes são as interações e as dependências das atividades dos diferentes agentes no contexto do Sistema Multi-Agente. Desta forma, a coordenação desempenha um papel essencial nos SMA´s pois tais sistemas são distribuídos.

Os SMA´s incluem diversos agentes que interagem ou trabalham em conjunto. Cada agente é basicamente um elemento capaz de resolver problemas de forma autônoma e operar assincronamente, em relação aos outros agentes. Para que um agente possa operar como parte do sistema é necessária a existência de uma infraestrutura que permita a comunicação e/ou interação entre os agentes que compõe o SMA.

Neste contexto surgem as organizações de agentes, que podem ser descritas em três dimensões: (i) a estrutural (o que os agentes podem fazer); (ii) a funcional (como os agentes podem fazer); (iii) e a deôntica (o que os agentes devem fazer). Além disso, apoia-se nas noções de abertura e de heterogeneidade. Uma organização em um SMA pode ser vista simplificadamente como um conjunto de restrições adotadas por um grupo de agentes para que possam atingir seus objetivos mais facilmente [Boissier 1993] [Sichman 2003].

Uma abordagem centrada em organizações permite a descrição explícita da organização em vários aspectos, considerando conceitos sociais como as instituições, os grupos (as comunidades), os papéis (funções, posições), as tarefas (atividades) e os protocolos de interação (estruturas de diálogo). Dessa forma, os agentes formam uma sociedade, uma entidade robusta e flexível, onde eles podem interagir uns com os outros, identificar as capacidades e necessidades de cada um e pedir ou realizar atividades. Os agentes de uma sociedade podem determinar que atividades que devem ser realizadas e envolver ainda outros agentes para os ajudarem. As sociedades fornecem a base para se construírem organizações mais estruturadas, ou seja, padrões de comportamento e interações que sejam estáveis e que mudem lentamente com o tempo.

As organizações têm estruturas com três dimensões relacionadas: (i) Poder, que define os padrões de delegação de tarefas; (ii) Coordenação, que diz respeito ao fluxo de conhecimento na organização e (iii) Controle, que está associada às funções de recuperação de tarefas na organização [Grossi 2006].

6.2. Sistemas Multi-Agentes Normativos

Uma norma social trata-se de um guia prescrito para conduta ou ação que geralmente é cumprido pelos membros da sociedade [Ullman-Margalit 1977]. Ela é criada para regular o comportamento das entidades no sistema, através do que é proibido, permitido ou obrigado e representa um papel fundamental no contexto organizacional.

Uma norma possui os seguintes aspectos:

- **Regulação de comportamento:** A norma, definida na sociedade, estabelece um comportamento esperado por parte de um agente em uma sociedade ou numa dada circunstância.
- **Mecanismos de sanção:** Quando um agente segue ou não uma norma, pode ser submetido a uma sanção. Tal sanção deverá ser de recompensa e/ou punição. A

sanção de punição pode incluir perda monetária, perda de utilidade ou ainda a diminuição da pontuação de reputação.

- **Mecanismos de divulgação:** Exemplos de mecanismos de divulgação da norma inclui a noção de líderes, imitação e aprendizagem por parte de um agente.

Sistemas Multi-Agente Normativos é um sistema organizado por meio de mecanismos criados para representar, comunicar, distribuir, detectar, criar, modificar e cumprir as normas. Além disso, apresentam mecanismos para deliberação sobre as normas e detecção de violação ou cumprimento de norma.

As normas ajudam na manutenção da ordem social e na previsão mais correta sobre o comportamento do agente na sociedade. Além disso, elas melhoram a cooperação e a colaboração reduzindo a computação necessária para a tomada de decisão [Epstein 2001]. No entanto, devido a sua autonomia, os agentes podem ter a tendência de não cumprir com as normas especificadas.

6.3 Modelo Organizacional MOISE+

MOISE + (Modelo de Organização para Sistemas Multi-Agentes) [Hübner 2002] é um modelo de organização que estabelece quais os componentes que formam uma organização e como estes podem contribuir para um SMA, restringindo comportamentos dos agentes através de uma estrutura de ligações entre os papéis e um conjunto de planos globais. Este modelo foi desenvolvido para ajudar o processo de reorganização, e por isso apresenta características que suportam tanto a avaliação como o projeto de novas organizações.

Este modelo apoia-se fortemente no modelo MOISE [Hannoun 2002] e apresenta uma visão centrada na organização considerando três formas de representar as restrições organizacionais (papéis, planos e normas). Este modelo possui duas noções centrais: uma especificação organizacional adotada por um grupo de agentes, os quais formam uma entidade organizacional; (ii) A ação relacionada a esta entidade organizacional se destina a atingir um objetivo.

Este modelo distingue explicitamente três aspectos para modelar uma organização:

- **Aspecto estrutural**: define as relações entre os agentes, com base nas noções de papel, grupo e vínculo social. Os papéis encapsulam funcionalidades, objetivos, planos e relações com outras papéis, que um agente deve seguir quando está em um grupo. Também define restrições cardinalidade, herança, compatibilidades e dependências entre os papéis.
- **Aspecto Funcional**: descreve os objetivos globais do sistema. Estes objetivos são decompostos em planos que, por sua vez, são distribuídos para os agentes como missões. Assim, uma missão representa um conjunto coerente de objetivos que são atribuídos aos papéis. Portanto, se um agente aceita uma missão, está empenhado em atingir todos os objetivos desta missão.
- **Aspecto Deôntico**: define as regras que regem o sistema, em termos de permissões, proibições e obrigações relacionadas com os diversos tipos de papéis. As regras também podem ser acompanhadas por sanções, aplicada em caso de cumprimento ou violação delas.

Agradecimento

Agradecemos à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - **CAPES** - Brasil pelo suporte financeiro concedido durante a produção deste trabalho.

Referências

Coelho, H. "Teoria da Agência: Arquitectura e Cenografia". Sesimbra: Portugal (2008).

Boissier O.; "Problema de controle em um sistema de visão integrada, utilizando um sistema multi-agente". Tese de doutorado, INP Grenoble, França (1993).

Bordini, R. H., Hübner, J. F., and Wooldridge M. "Programming Multi-Agent Systems in AgentSpeak using Jason". Series Editor: University Liverpool (2007).

Bordini, R. H., Vieira, R. "Linguagens de Programação Orientadas a Agentes: uma introdução baseada em AgentSpeak(L)". Revista de informática teórica e aplicada: Porto Alegre. Vol.10, n.1 (2003).

Epstein J. M. , "Learning to be thoughtless: Social norms and individual computation", Computational Economics (2001).

Grossi, D., Dignum, F. e Royakkers, L. "Structural Evaluation of Agent Organizations", Proceedings of AAMAS'06, pp. 1110-1112 (2006).

Jennings, N. R. ,Wooldridge, M. (eds.). "Agent technology: foundations, applications and markets." Springer Verlag,(1998).

Jennings, N. R. "On agent-based software engineering". Artificial intelligence 117.2 (2000): 277-296.

Lesser, Victor, "Cooperative Multi-Agent Systems: A Personal View of the State of the Art". IEEE Transactions on Knowledge and Data Engineering, Vol. 11, Nº 1. (1999).

Odell, J., Parunak, H. V. D., Fleischer, M., and Brueckner, S. (2003). "Modeling agents and their environment", In Agent-oriented software engineering III, volume 2585 of LNCS (pp. 16–31). Berlin/Heidelberg: Springer.

Pressman, R. S. "Engenharia de Software". São Paulo: Makron Books, 1995.

Rao, A. S. "AgentSpeak (L): BDI agents speak out in a logical computable language." Agents Breaking Away. Springer Berlin Heidelberg, (1996).

Ricci, A., Piunti, M. and Viroli, M., "Environment programming in multi-agent systems: an artifcat-based perspective", In: proccedings of the Autonomous Agent Multi-Agent Systems, 2011. Berlin, Springer.

Russell, S. and Norvig, P. "Artificial Intelligence: A Modern Approach". 3 ed. Prentice-Hall (2009).

Sichman J. S.; "Raciocínio Social e Organizacional em Sistemas Multiagentes: Avanços e perspectivas". Tese de Doutorado. Universidade de São Paulo. (2003)

Sommerville, I. Engenharia de Software. São Paulo: Pearson Prentice Hall, 2011.

Ullmann-Margalit E., "The Emergence of Norms", Clarendon Press (1977).

Wooldridge, M. "An introduction to multiagent systems". Wiley (2008).

Introdução a Redes Neurais Artificiais com a biblioteca Encog em Java

Raquel Machado de Sousa[1]

[1]Laboratório de Sistemas Inteligentes (LSI) – Universidade Federal do Maranhão (UFMA)
Av. dos Portugueses – São Luis – MA – Brasil
rachel.msousanet@gmail.com

Abstract. *This article describes basic concepts of artificial neural networks, ranging from its history of development that spurred the research in this area until the main features involved in the construction of an ANN. Being demonstrated through the development framework Encog used to build RNAs in java, which provides a range of possible application in various sectors.*

Resumo. *Este artigo descreve os conceitos básicos de redes neurais artificiais, que vão desde seu histórico de desenvolvimento que impulsionaram as pesquisas nesta área, até as principais características envolvidas na construção de uma RNA. Sendo demonstrado através do framework de desenvolvimento Encog utilizada para construção de RNAs em java, que oferece uma gama de possibilidade de aplicação em vários setores.*

1. Introdução

As redes neurais artificiais (RNAs) são uma tecnologia que têm sido utilizada para diversos fins, baseada no funcionamento dos neurônios biológicos, possui grande capacidade de aprendizado e generalização diante de exemplos expostos, mesmo com a interferência de ruídos. Conhecida como máquinas de processamento paralelo e por ser uma técnica de estatística não-linear, pode resolver uma grande variedade de problemas.

Segundo Haykin (2001) as RNAs são um processador maciçamente paralelamente distribuído constituído de unidades de processamento simples, que têm a propensão natural para armazenar conhecimento experimental. Podemos dizer que o processamento é paralelo no sentido que todos os neurônios dentro do conjunto ou camada processam as suas entradas simultaneamente e independentemente.

As redes neurais fazem parte de uma área de pesquisa conhecida como Inteligência Artificial, mais precisamente na abordagem conexionista, na qual utiliza de métodos computacionais que possuem e utilizam capacidade racional como os seres humanos na resolução de problemas.

Pelo seu caráter multidisciplinar está nas mais diversas aplicações nas quais podemos citar: avaliação de imagens captadas por satélite, classificação de padrões de escrita e fala, reconhecimento de faces em visão computacional, controle de trens de grande velocidade, previsão de ações no mercado financeiro, identificação de anomalias em imagens médicas. Diante da grande variedade de sua utilização, as tarefas mais comuns realizadas são classificação, regressão numérica, agrupamento, predição e reconhecimento de padrões.

2. Histórico das Redes Neurais Artificiais

Os trabalhos em redes neurais iniciaram no ano de 1943 com a publicação de um artigo por McCulloch & Pitts, no qual os autores realizaram o primeiro modelo matemático inspirado no neurônio biológico, resultando assim na primeira concepção de neurônio artificial. [Silva 2010]

Em 1949, é modelado o primeiro método de treinamento para redes neurais, chamado de regra de aprendizado de Hebb. Ele mostrou que a plasticidade da aprendizagem em redes neurais é conseguida através da variação dos pesos de entrada dos neurônios, propondo uma teoria para explicar o aprendizado em neurônios biológicos baseada no reforço das ligações sinápticas entre os neurônios excitados. [Braga 2007]

Desde então diversos outros pesquisadores continuaram desenvolvendo pesquisas fundamentadas no neurônio biológico e algoritmos de aprendizado. Surgiram modelos da mente, como o Perceptron, de Rosenblatt, em 1958, e o Adaline, de Bernard Widrow, em 1960.

A capacidade do Perceptron de reconhecer padrões simples trouxe muito interesse de pesquisadores para área neurocomputacional, mas em 1969, Minsky e Papert chamaram a atenção para algumas tarefas que o perceptron descrito por Rosenblatt não era capaz de executar. [Braga 2007] Por estar limitado a resolução de problemas linearmente separáveis, o perceptron não é capaz de resolver “problemas difíceis de aprender” como paridade, simetria e conectividade.

Devido a estas descobertas a abordagem conexionista ficou adormecida até meados dos anos 80 quando ressurgiram novamente pelas descobertas feitas por Hopfield com sua abordagem de energia em 1982 e o algoritmo de aprendizagem backpropagation para redes perceptron múltiplas camadas (*multilayer feed-forward networks*) primeiramente proposto por Werbos, reinventada diversas vezes e popularizada por Rumelhart et al. em 1986. [Jain, Mao e Mohiuddin 1996]

O marco histórico das principais publicações na área de RNAs é apresentado no quadro abaixo:

Tabela 1. Marco histórico da pesquisa em redes neurais artificiais

Ano	Autores
1943	McCulloch e Pitts
1948	Wiener
1949	Hebb
1957	Rosenblatt
1958	Widrow e Hoff
1969	Minsky e Papert
1960 - 1980	Kohonen, Grossberg, Widrow, Arderson, Caianiello, Fukushima e Igor Aleksander
1974	Werbos
1982	Hopfield
1986	Rumelhart e McClelland

3. Neurônio Biológico x Neurônio Artificial

3.1. Neurônio Biológico

O centro do sistema nervoso humano é o cérebro, representado pela rede neural (nervosa), que recebe continuamente informação, percebe e toma decisões apropriadas. [Haykin 2001] Através dos impulsos elétricos, também chamadas de sinapses, que transportam os sinais de informação, o cérebro tem o poder de atuar e responder a estímulos a todo momento.

O cérebro é formado por cerca de 10^{11} neurônios, cada um destes processa e se comunica com milhares de outros continuamente e paralelamente. Os neurônios biológicos são divididos, de maneira simplificada, em três seções: o corpo celular, os dendritos e o axônio, cada um com funções específicas, porém complementares. [Braga 2007]

Os dendritos são filamentos prolongados responsáveis pela captação, de forma contínua, dos estímulos vindos de diversos outros neurônios. O corpo celular recebe os estímulos advindos dos dendritos, onde é processado as informações, a fim de produzir um potencial de ativação que indicará se o neurônio poderá disparar um impulso elétrico ao longo do axônio. O axônio que é formado por um único prolongamento, têm por função conduzir os impulso elétricos para os outros neurônios conectores chegando até os dendritos.

As sinapses são as unidades estruturais que se configuram como as conexões que viabilizam a transferência de impulsos elétricos do axônio de um neurônio para os dendritos dos outros. [Silva 2010] Podemos dizer que uma sinapse converte um sinal elétrico pré-sináptico em um químico e então de volta, em um sinal elétrico pós-sináptico. Na figura abaixo podemos visualizar o modelo simplificado do neurônio biológico e a propagação dos sinais sinápticos.

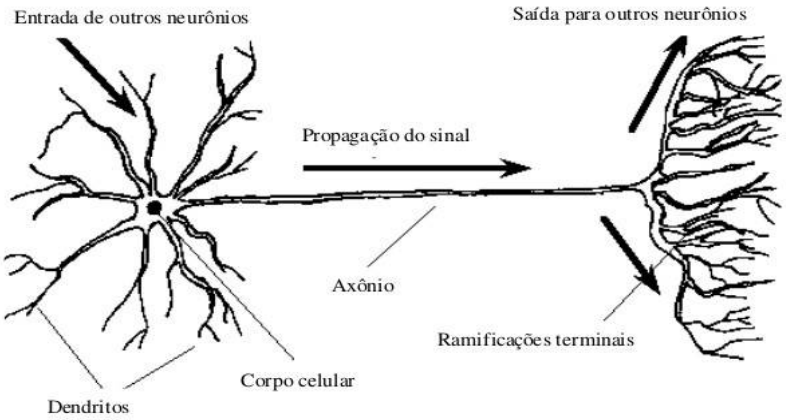


Figura 1. Modelo da célula neural biológica

3.2. Neurônio Artificial

A analogia entre o comportamento do neurônio biológico e o neurônio artificial foi o que possibilitou e deu base para as pesquisas em redes neurais. Fazendo a correlação com o modelo do neurônio mais simples de McCulloch & Pitts (1943), os sinais de entrada representados pelo conjunto $\{ x_1, x_2, \ldots, x_n \}$, equivalem aos impulsos elétricos

externos captados pelos dendritos.

As ponderações exercidas pelas junções sinápticas do modelo biológico são representadas no neurônio artificial pelo conjunto de pesos sinápticos $\{ w_1, w_2, \ldots, w_n \}$. Obtendo - se a soma ponderada dos sinais de entrada $\{ x_n \}$ e seus respectivos pesos sinápticos $\{ w_n \}$ obtemos nosso potencial de ativação, denotado por u, e representado no neurônio biológico pelo corpo celular que gera o sinal de saída propagado pelo axônio. Esse sinal de saída é representado por y no neurônio artificial.

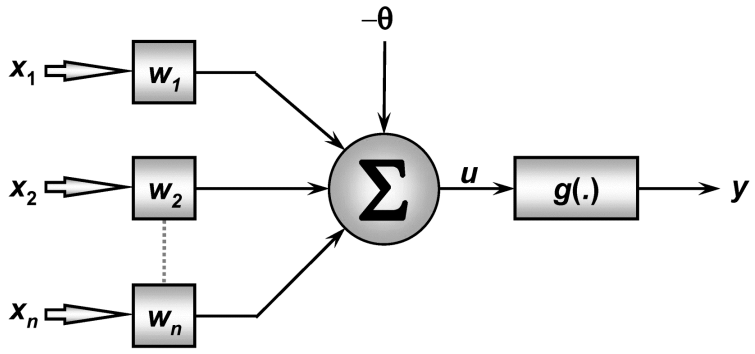


Figura 2. Modelo de um neurônio artificial

Considerando a figura 2, identifica-se os principais elementos que constituem o neurônio artificial, que de acordo com Silva (2010) são 7:

- Sinais de entrada $\{ x_1, x_2, \ldots, x_n \}$:

  São sinais advindos do meio externo e que representam os valores assumidos pelas variáveis de uma aplicação específica. Esses sinais geralmente são normalizados visando melhorar a eficiência computacional dos algoritmos de aprendizagem;

- Pesos sinápticos $\{ w_1, w_2, \ldots, w_n \}$

  São os valores que servirão para ponderar cadas uma das variáveis de entrada da rede, permitindo-se quantificar a sua relevância em relação à funcionalidade do respectivo neurônio;

- Combinador linear $\{ \sum \}$:

  Sua função é agregar todos os sinais de entrada e pondera-los com os respectivos pesos sinápticos a fim de produzir um valor de potencial de ativação;

- Limiar de ativação $\{\theta\}$:

  É uma variável que especifica qual será o patamar apropriado para que o resultado produzido pelo combinador linear possa gerar um valor de disparo em direção a saída do neurônio;

- Potencial de ativação $\{ u \}$:

  É o resultado produzido pela diferença do valor produzido entre o combinador linear e o limiar de ativação. O resultado de u é representado por:

$$u = \sum_{i=0}^{n} w_i . x_i - \theta$$

- Função de ativação { g }:

  A função limita a saída do neurônio dentro de um intervalo de valores razoáveis a serem assumidos pela sua própria imagem funcional.

- Sinal de saída { y }:

  É o valor final produzido pelo neurônio em relação a um determinado conjunto de sinais de entrada, podendo ser também utilizado por outros neurônios que estão sequencialmente interligados. $y = g(u)$

4. Funções de Ativação

Como já citado acima a função de ativação é responsável por gerar uma saída $y = g(u)$ a partir dos valores de entrada e seus respectivos pesos. Dependendo do domínio empregado as redes devem apresentar uma não – linearidade em sua saída, que seja suave, portanto, diferenciável. Então, quando propriedades dinâmicas estão envolvidas na definição do estado de ativação, equações diferenciais ou as diferenças são empregadas. [Silva 1998]

As principais funções utilizadas são [Silva 2010]:

a) Função linear

Também chamada de função identidade, a função linear, produz resultados de saída idênticos aos valores do potencial de ativação { u }, sua expressão matemática é definida por:

$$g(u) = u$$

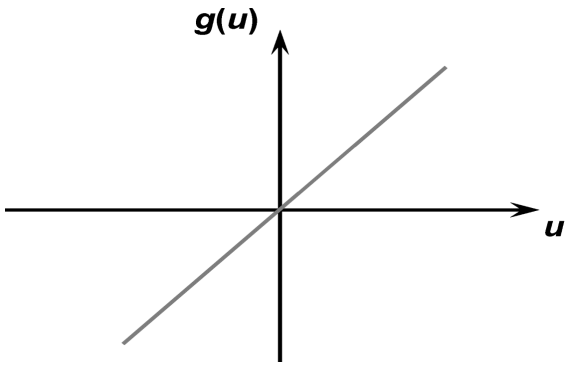


Figura 3. Função de ativação linear

b) Função degrau

O resultado produzido assumirá valores unitários positivos quando o potencial de ativação for maior que ou igual a zero e valores nulos caso seja o potencial seja menor que zero.

$$g(u) = \begin{cases} 1, se\ u \geqslant 0 \\ 0, se\ u < 0 \end{cases}$$

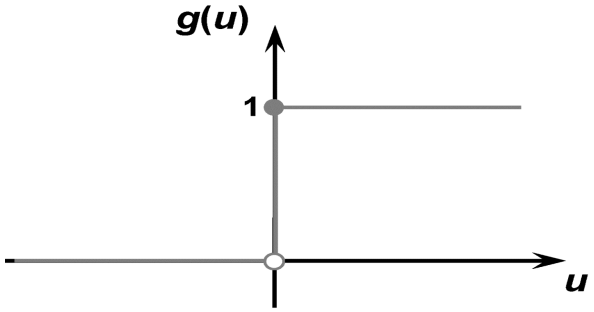


Figura 4. Função de ativação degrau

c) Função degrau bipolar

O resultado desta função assumirá valores unitários positivos quando o potencial de ativação do neurônio for maior que zero, valor nulo quando o potencial for nulo e valores negativos quando o potencial for menor que zero.

$$g(u)=\begin{Bmatrix} 1, se\ u>0 \\ 0, se\ u=0 \\ -1, se\ u<0 \end{Bmatrix}$$

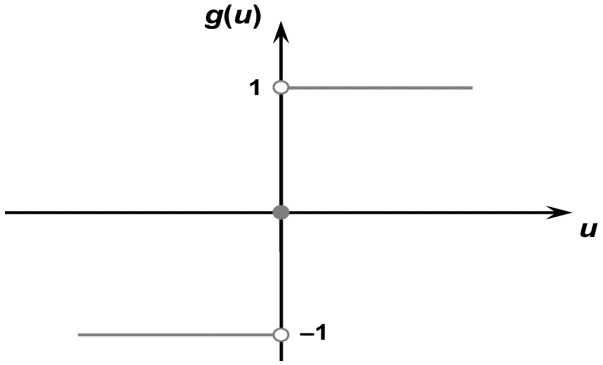


Figura 5. Função de ativação degrau bipolar

d) Função logística

O resultado de saída produzido pela função logística assumirá valores entre zero e um, de acordo com a seguinte expressão:

$g(u)=\frac{1}{1+e^{-\beta . u}}$, onde β é uma constante real associada ao nível de inclinação da função logística frente ao seu ponto de inflexão.

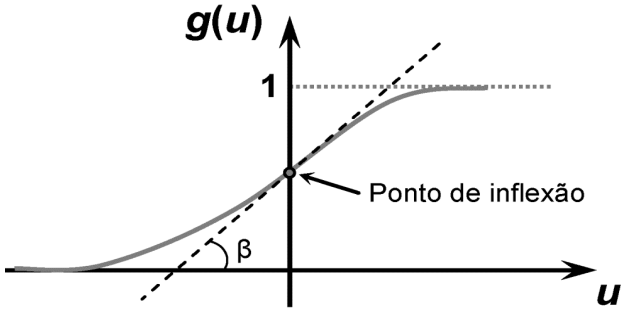


Figura 6. Função de ativação logística

e) Função tangente hiperbólica

O resultado de saída sempre assumirá valores reais entre -1 e 1, cuja expressão matemática é definida por:

$g(u)=\frac{1-e^{-\beta.u}}{1+e^{-\beta.u}}$, β aqui também está associado ao nível de inclinação em relação ao ponto de inflexão da função.

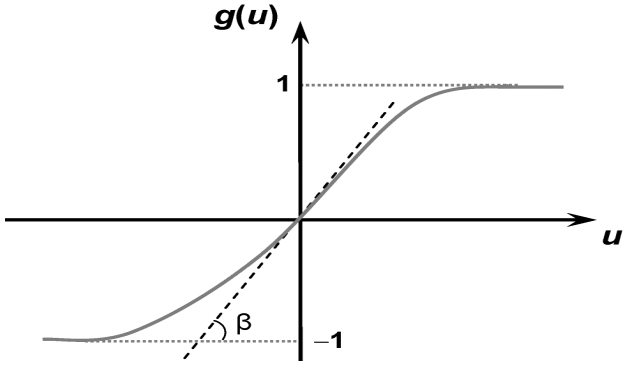


Figura 7. Função de ativação tangente hiperbólica

f) Função gaussiana

A saída da função gaussiana produzirá resultados iguais para aqueles valores de potencial de ativação $\{u\}$ que estejam posicionados a uma mesma distância do seu centro(média), sendo que a curva é simétrica em relação a este. A função é dada pela seguinte equação :

$g(u)=e^{-\frac{(u-c)^2}{2\sigma^2}}$, onde c é um parâmetro que define o centro da função gaussiana e σ denota o desvio padrão associada a mesma.

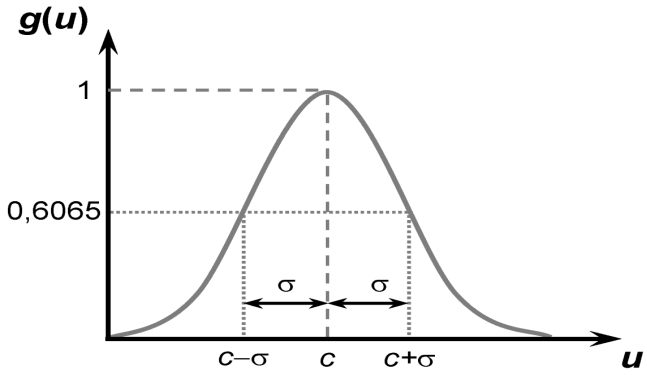


Figura 8. Função de ativação gaussiana

5. Estruturas de RNAs

A maneira pela qual os neurônios de uma rede neural são estruturados está intimamente ligada com o algoritmo de aprendizagem usado para treinar a rede. Podemos, portanto, falar de algoritmos de aprendizagem utilizados no projeto de redes neurais como sendo estruturados. [Haykin 2001]

Assim, o desempenho de uma rede neural depende não só da função de ativação

usada pelos neurônios, mas também de como esses neurônios são arranjados, isto é, estruturados uns com os outros, e dos algoritmos de aprendizado utilizados no processo de treinamento da rede.

Jain, Mao e Mohiuddin (1996) consideram apenas dois tipos de arquiteturas de RNA: redes *feedforward* e redes recorrentes. Haykin (2001) e Silva (2010) dividem a arquitetura de uma RNA em três tipos: redes *feedforward* de camada simples, redes *feedforward* de camada múltiplas e redes recorrentes. Aqui consideraremos estes três tipos de arquitetura.

5.1. Redes *feedforward* de camada simples

Este tipo de estrutura é formada por apenas uma camada de entrada e uma única camada de neurônios, que é a mesma camada de saída. Esta rede é estritamente do tipo alimentada adiante ou acíclica, e é chamada de rede de camada única, sendo que a designação "camada única" se refere à camada de saída de nós computacionais (neurônios). Não contamos a camada de entrada de nós, porque lá não é realizada qualquer computação. [Haykin 2001]

Exemplos de redes desse tipo de arquitetura são as redes Perceptron e a Adaline.

Figura 9. Rede feedforward de camada simples

5.2. Redes *feedforward* de camada múltipla

As redes *feedforward* de camadas múltiplas são constituídas pela presença de uma ou mais camadas escondidas de neurônios. Com os neurônios ocultos é possível intervir entre a camada de entrada e a saída da rede de uma maneira útil tornando a rede capaz de extrair estatísticas de ordem elevada. Como exemplos desse tipo de rede temos as Perceptron multicamadas (*multilayer Perceptron – MLP*) e as redes de base radial (*radial basis function – RBF*)

Figura 9. Rede feedforward de camada múltipla

5.3. Redes recorrentes

São redes em que as saídas dos neurônios são realimentadas como sinais de entrada para outros neurônios. Estas redes possuem uma característica de realimentação que as qualifica para o processo dinâmico de informações.[Silva 2010]

Um exemplo para esse tipo de rede são as redes Hopfield, que pode ser ilustrado na figura.

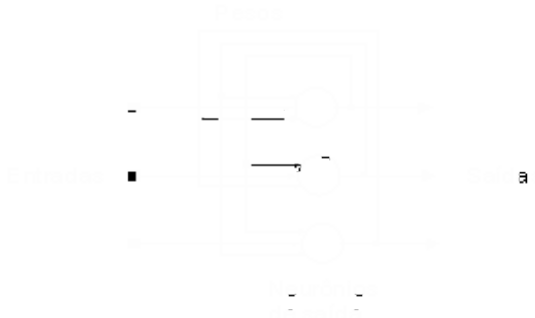


Figura 10. Rede recorrente

6. Aprendizado em Redes Neurais

Aprendizagem é uma das características mais relevantes no processo de treinamento de uma rede neural. A definição geral de aprendizagem segundo Haykin (2001) é expressa da seguinte forma:

"A aprendizagem é um processo pelo qual os parâmetros livres de uma rede neural são adaptados através de um processo de estimulação pelo ambiente no qual a rede está inserida. O tipo de aprendizagem é determinado pela maneira pela qual a modificação dos parâmetros ocorre."

A aprendizagem ocorre no momento que a rede é estimulada pelo seu ambiente, de forma a sofrer modificações nos seus parâmetros livre, isto é, seus pesos sinápticos, como resultado desta estimulação; e por fim, responde de uma nova maneira ao ambiente por causa da modificação ocorrida na sua estrutura interna.

O treinamento ou aprendizado de um RNA ocorre após a preparação dos dados e da escolha do modelo e da arquitetura de RNA a serem utilizados para os dados. O treinamento de uma RNA não possui tempo determinado de execução e, na maioria dos casos, acontece com um subconjunto de exemplos. [Barone 2003]

Há vários algoritmos diferentes para treinamento de redes neurais, Haykin (2001) e Braga (2007) agrupam em dois paradigmas principais: aprendizado supervisionado e aprendizado não – supervisionado. Jain, Mao e Mohiuddin (1996) considera três grupos: aprendizado supervisionado, aprendizado não-supervisionado e aprendizado híbrido. Aqui consideraremos a literatura de Haykin para esta classificação. Apesar de ainda não se chegar a um consenso se a aprendizagem híbrida ou aprendizagem por reforço faz parte do paradigma supervisionado ou não-supervisionado.

6.1. Aprendizado supervisionado

No aprendizado supervisionado é considerado em cada amostra sinais de entrada e sinais de saída desejadas no processo. Também chamada de aprendizagem com professor, pois podemos considerar o professor como tendo conhecimento sobre o

ambiente, sendo o conhecimento representado pelo conjunto de exemplos de entrada-saída.

Segundo Haykin (2001) o professor é capaz de fornecer à rede neural uma resposta desejada, que representa a ação ótima a ser realizada pela rede neural. Os parâmetros da rede são ajustados sob a influência combinada do vetor de treinamento e do sinal de erro. O sinal de erro é definido como a diferença entre a resposta desejada e a resposta real da rede. O ajuste é então realizado com o objetivo da rede atingir a solução ótima no sentindo estatístico. Quando a condição é alcançada, o professor deixa de atuar sobre a rede neural e lida por si mesma com o ambiente.

6.2. Aprendizado não – supervisionado

O aprendizado não-supervisionado, chamado também de aprendizagem sem professor, somente os padrões de entrada estão disponíveis para a rede, ao contrário do aprendizado supervisionado, cujo o conjunto de treinamento possui pares de entrada e saída. [Braga 2007]

Haykin (2001) divide esse paradigma em ainda duas seções:

a) Aprendizagem por reforço

   Em muitas literaturas este tipo de aprendizagem é considerada como um tipo de aprendizagem supervisionada ou como um terceiro paradigma de aprendizagem. A aprendizagem por reforço é realizado através de um crítico externo que procura maximizar o reforço das boas ações executadas pela rede.

   O processo de treinamento da rede é realizado tipicamente por tentativa e erro, pois a única resposta disponível para determinada entrada é se esta é satisfatória ou não. Se for considerada satisfatória, incrementos nos pesos sinápticos e limiares são então gradualmente efetuados visando recompensar esta condição comportamental. [Silva 2010]

b) Aprendizagem não-supervisionada

   Neste tipo de aprendizagem não existe qualquer professor ou crítico que ajude no treinamento da rede. Em vez disso, são dadas condições para a rede se auto organizar, de acordo com uma medida de representação e os parâmetros livres da rede são otimizados em relação a esta medida.

   Durante processo de aprendizado os padrões de entrada são apresentados continuamente à rede, e a existência de regularidades nesses dados faz com que o aprendizado seja possível. Regularidade e redundância nas entradas são características essenciais para haver aprendizado não-supervisionado. [Braga 2007]

7. Regras de Aprendizagem

As regras de aprendizagem definem como ocorrerá o ajuste dos pesos entre os sucessivos ciclos de treinamento de uma rede neural. [Basheer e Hajmeer 2000]

Os tipos básicos de regra de aprendizagem são: aprendizagem por correção de erro; aprendizagem baseada em memória; aprendizagem hebbiana; aprendizagem competiviva e a aprendizagem de boltzman.

A aprendizagem por correção de erro utiliza o paradigma de aprendizagem

supervisionada. Seu princípio básico é usar o sinal de erro para modificar os pesos a fim de reduzir o erro gradualmente. A regra de aprendizagem do perceptron e do backpropagation é baseado neste princípio. [Jain, Mao e Mohiuddin 1996]

Na aprendizagem baseada em memória todas as experiências passadas são armazenadas explicitamente em uma grande memória de exemplos de entrada-saída. O método utilizado é o método do vizinho mais próximo.

A aprendizagem hebbiana trouxe uma grande avanço para a área de redes neurais, nesta aprendizagem se dois neurônios em ambos os lados de uma sinapse são ativados simultaneamente, então a força daquela sinapse é seletivamente aumentada. Se dois neurônios em ambos os lados de uma sinapse são ativados assincronamente, então aquela sinapse é seletivamente enfraquecida ou eliminada. [Haykin 2001]

Quando falamos em aprendizagem competitiva, a ideia é que, neste caso, dado um padrão de entrada, fazer com que as unidades de saída disputem entre si para serem ativadas. Existe uma competição entre as unidades de saída para decidir qual delas será vencedora e, consequentemente terá sua saída ativada. A unidade vencedora tem seus pesos atualizados no treinamento. [Braga 2007]

A aprendizagem de boltzman é um algoritmo de aprendizagem estocástico derivado de ideias enraizadas na mecânica estatística. Uma rede neural projetada com base na regra de aprendizagem de boltzman é denominada uma máquina de boltzman.

O objetivo de boltzman aprendizagem é ajustar os pesos de conexão para que os estados das unidades visíveis satisfaçam uma determinada distribuição de probabilidade desejada. [Jain, Mao e Mohiuddin 1996]

8. Construindo RNAs em java com Encog

O Encog é um framework da Inteligência Artificial para Java que suporta não só redes neurais como outras áreas da IA. É uma estrutura de aprendizagem de máquina avançada que suporta uma variedade de algoritmos avançados, bem como métodos de apoio para normalizar e processar dados. Algoritmos de aprendizagem de máquina, como Support Vector Machines, Redes Neurais Artificiais, programação genética, redes Bayesianas, Hidden Markov modelos e algoritmos genéticos são suportados. [Heaton 2011]

8.1. Instalação do Encog

Para instalar é necessário baixar a última versão do Encog no seguinte link:

http://www.heatonresearch.com/encog/

Depois é só extrair os seguintes arquivos em um local desejado e incluir o caminho das bibliotecas ao usar uma IDE.

- The Encog Core
- The Encog Examples
- The Encog Workbench

Para mais informações acesse: http://www.heatonresearch.com/wiki/Getting_Started

8.2. Construindo uma rede neural

As redes neurais no Encog são constituídas por camadas, sinapses, propriedades e uma classe lógica neural. Uma camada é uma coleção de neurônios semelhantes; a sinapse liga uma camada a outra; as propriedades de uma rede neural definem as qualidades únicas que um tipo de rede neural pode ter; a classe lógica neural define como a saída da rede neural deve ser calculada; e as funções de ativação medem a saída de uma camada antes de atingir a camada seguinte.

Para exemplificarmos, usaremos o problema do XOR na criação de uma rede neural simples. Aqui usaremos uma rede feedforward com o algoritmo backpropagation para treinamento da rede neural.

O algoritmo de treinamento backpropagation segue os seguintes passos para ajuste dos pesos:

1. Apresentação de um padrão X à rede, o qual fornece uma saída Y;

2. Cálculo do erro (diferença entre o valor desejado e a saída) para cada saída;

3. Determinação do erro retropropagado pela rede associado à derivada parcial do erro quadrático.

4. Ajuste dos pesos de cada elemento;

5. Por fim, um novo padrão é apresentado a rede e o processo é repetido até que ocorra a convergência, ou seja, (erro < tolerância estabelecida) ou o número de interações corresponda a um valor máximo estabelecido.

No Encog é preciso definir o vetor de entrada e de saída desejada da rede neural, a camada de entrada da rede é definida por um vetor *double* onde o tamanho significa a quantidade de neurônios.

```
//Entrada necessária para o XOR
public static double XOR_INPUT[][] = {{1.0,0.0}, {0.0,0.0}, {0.0,1.0},
{1.0,1.0}};
// Dados ideais necessários para XOR
public static double XOR_IDEAL[][] = {{1.0}, {0.0}, {1.0}, {0.0}};
```

Depois de definido os dados do treinamento, uma rede neural básica é criada através das classes *BasicNetwork* e *BasicLayer.*

```
//Cria a rede neural
BasicNetwork network = new BasicNetwork();
network.addLayer(new BasicLayer(null,true,2));
network.addLayer(new BasicLayer(new ActivationSigmoid(), true, 2));
network.addLayer(new BasicLayer(new ActivationSigmoid(),true,1));
network.getStructure().finalizeStructure();
network.reset();
```

Aqui na estrutura da rede neural chamada *network* são adicionados a camada de entrada com 2 neurônios, uma camada oculta com 2 neurônios e a camada de saída com 1 neurônio, através do método *addLayer*. Por fim, é informada a estrutura da rede que não serão mais adicionados camadas na rede através do método *finalizeStructure*. O

método *reset* torna aleatório os pesos das redes entre as camadas.

O treinamento é feito através da classe *MLDataSet*, o objeto irá conter o conjunto de treinamento. A implementação do treinamento então é feito através do algoritmo Backpropagation que pode ser implementado da mesma maneira pela interface *Train.* Em seguida é inicializado o número de épocas e a interação de treinamento da rede. O teste da rede neural é realizado pela classe *MLDataPair* que contém os pares de entrada e saída do treinamento. A classe *MLData* computa a saída da rede, dado a rede treinada e os respectivos dados de entrada somente. Por fim , são exibidos os dados de saída ideal e os dados reais recebido pelo objeto *MLData.*

```
//Cria dados de treinamento
MLDataSet trainingSet = new BasicMLDataSet(XOR_INPUT, XOR_IDEAL);
//Treinamento da rede neural
final Backpropagation train = new Backpropagation(network,trainingSet,
0.7, 0.3);
//Inicializo o numero de épocas
int epoch = 1;

//Interação de treinamento da rede
do {
  train.iteration();
  System.out.println("Epoch #" + epoch +"Error: " + train.getError());
  epoch++;
} while(train.getError() > 0.01);

//Teste da rede neural
System.out.println("Neural Network Results:");
for (MLDataPair pair: trainingSet){
   final MLData output = network.compute(pair.getInput());
   System.out.println(pair.getInput().getData(0) + ","
                      + pair.getInput().getData(1) + ", actual="
                      + output.getData(0) + ", ideal="
                      + pair.getIdeal().getData(0));
}
Encog.getInstance().shutdown();
```

Referencias

Basheer, I. A., & Hajmeer, M. (2000). Artificial neural networks: fundamentals, computing, design, and application. *Journal of microbiological methods*, *43*(1), 3-31.

Barone, D.A.C. (2003), Sociedades artificiais: a nova fronteira da inteligência nas máquinas, Bookman.

Braga, A. P. (2007), Redes neurais artificiais: teoria e aplicações, LTC, 2ª edição.

Jain, A. K., Mao, J., & Mohiuddin, K. M. (1996). Artificial neural networks: A tutorial. *Computer*, *29*(3), 31- 44.

Heaton, J. (2011), Programming Neural Networks with Encog3 in Java, Heaton Research.

Haykin, S. (2001), Redes neurais: princípios e prática, Bookman, 2ª edição.

Silva, I.N. (2010), Redes neurais artificiais: para engenharia e ciências aplicadas, Artliber.

Silva, L.N.C. (1998). "Análise e síntese de estratégias de aprendizado para redes neurais artificiais." Dissertação (Mestrado em Engenharia Elétrica). Universidade Estadual de Campinas.

Tafner, M.A. (1999). "Estagiamento Automático do Sono Utilizando um Canal de EEG e uma Rede Neural Artificial com Alta Representação Cortical." Tese (Doutorado em Engenharia de Produção). Universidade Federal de Santa Catarina, http://www.eps.ufsc.br/teses99/tafner/, Outubro.

Utilizando Java para Construir e Destruir Robôs

Luís Bruno Pereira do Nascimento[1], Darielson Araújo de Souza[1], George Max Pereira de Souza[1].

[1] Universidade Estadual do Piauí (UESPI)
Caixa Postal 64200-000 – Parnaíba – PI – Brasil

{luisbrunu,daryewson,georgemaxphb}@gmail.com

Abstract. *The work proposed here focuses on the presentation Robocode, a tool used by many institutions as a teaching aid in the learning of object-oriented programming, Java language and helping in understanding (in practice) the concept of sensors and actuators within Intelligence artificial. The purpose of the tool is part of the improvement process of programming logic, where the player implements its own robots and perfect for battle against other virtual robots, thus bringing a greater understanding of programming logic, a larger field of object-oriented paradigm and excellent work to develop reasoning kidding.*

Resumo. *O trabalho ora proposto tem como foco apresentar o Robocode, uma ferramenta utilizada por diversas instituições de ensino como um auxílio no aprendizado da programação orientada a objetos, da linguagem Java e ajudando na compreensão (na prática) do conceito de sensores e atuadores dentro de Inteligencia Artificial. O intuito da ferramenta é fazer parte do processo de aperfeiçoamento da lógica de programação, onde o jogador implementa seus próprios robôs e o aperfeiçoa para duelar contra outros robôs virtuais, trazendo assim um maior entendimento da lógica de programação, um maior domínio do paradigma orientado a objetos e um excelente trabalho para desenvolver o raciocínio brincando.*

1. Introdução

Java é uma linguagem de programação orientada a objetos tida como madura, porém sua curva de maturidade se encontra em um constante processo de crescimento. Ela é uma linguagem multiplataforma, pois seus softwares podem ser executados em vários sistemas operacionais, basta ter sua máquina virtual, a JVM, compondo assim a plataforma Java.

Segundo a Oracle, o Java é executado em mais de 850 milhões de computadores pessoais e em bilhões de dispositivos em todo o mundo, inclusive telefones celulares e dispositivos de televisão. Isso mostra a importância de aprender dessa linguagem nesse meio tecnológico.

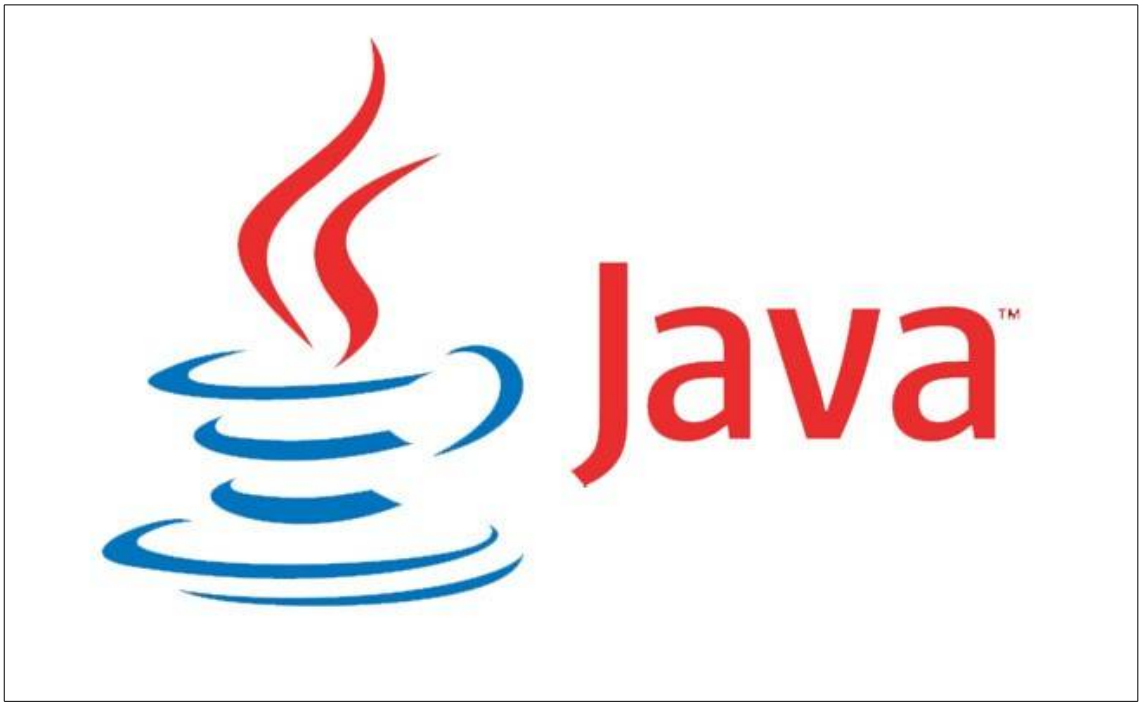


Figura 1. Logo oficial Java

A comunidade de usuários Java têm criado muitas ferramentas para auxiliar nesse processo de aprendizado, e o Robocode é um exemplo disso, pois ele é um jogo onde o objetivo é implementar um pequeno robô para a batalhar contra outros robôs, criando métodos para aperfeiçoar o poder de percepção e ação do seu robô, aperfeiçoando também seu raciocínio lógico e sua prática em programação orientada a objetos.

2. Histórico de Java com Robôs

Como foi afirmado acima, Java não é usado apenas como linguagem para desenvolvimento de softwares desktop, pode-se tomar como exemplo o projeto LeJOS[1], um trabalho open-source que possibilitou o Java em controladores RCX.

Um outro projeto bem interessante, que na verdade se trata de uma “curiosidade” para muitas pessoas, onde alguns robôs que estão em Marte possuem JVM´s embutidas com real time Java.

[1] www.lejos.org

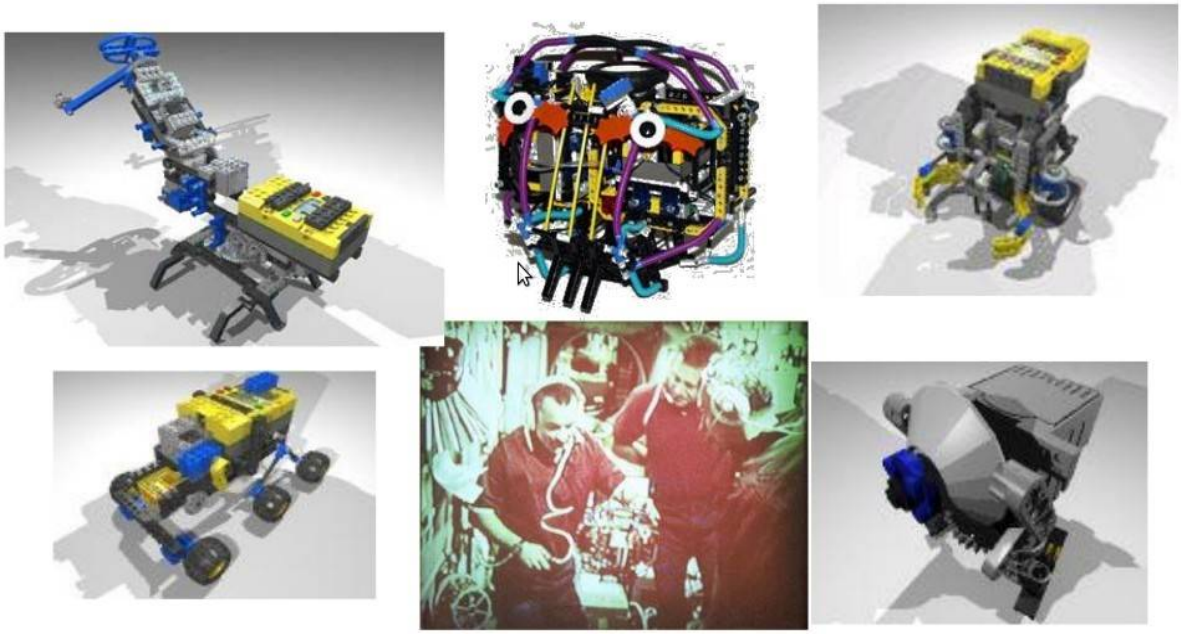


Figura 2. LeJOS(Java em Lego).

3. O Robocode

O jogo foi iniciado por Matthew A. Nelson no final de 2000 e se tornou profissional quando ele a trouxe à IBM, na forma de um download AlphaWorks, em julho de 2001.

Figura 3. Robocode.

O Robocode foi colocado no SourceForge[2] no início de 2005 como uma ferramenta de código aberto para ser continuado, pois estava com o desenvolvimento parado.

[2] http://robocode.sourceforge.net/

A comunidade começou a desenvolver suas próprias versões para eliminar os erros que haviam na versão existente, e assim foi crescendo com novas funcionalidades. Após o seu auge, mais de um ano se passou e o projeto não tinha mais mudança, assim Flemming Larsen N. assumiu o projeto Robocode no SourceForge como administrador e desenvolvedor em julho de 2006, incorporando as várias contribuições da comunidade Robocode surgindo assim a versão 1.1. Desde então, novas versões foram lançadas com mais recursos e novas contribuições da comunidade.

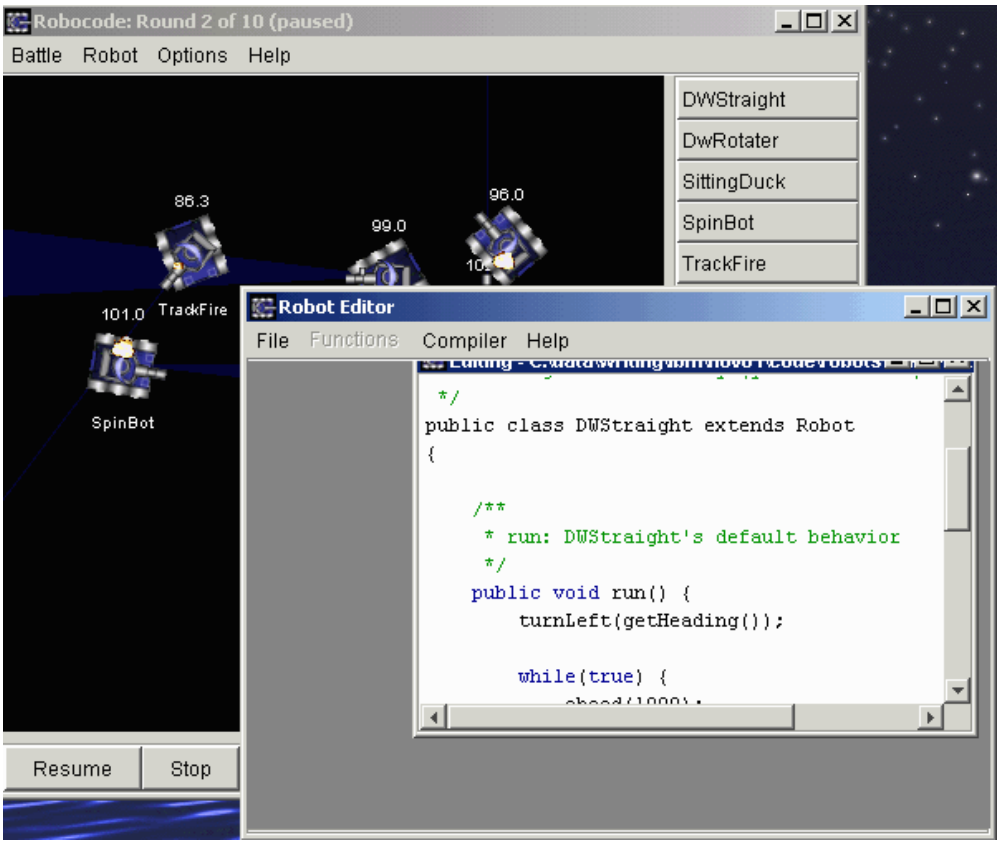


Figura 4. Robocode em uma de suas versões mais antigas.

Atualmente o Robocode está na versão oficial (instavel) 1.8.3.0 de 04 de Outubro de 2013, podendo programar os robôs em .NET além de java, desde a versão 1.7.2.0.

4. Conceitos básicos

Antes de o jogador iniciar o programa é importante atentar para alguns conceitos como método, atributo e evento.

O método é o elemento que representa uma chamada para algum procedimento de um objeto. O atributo é a característica atribuída a um objeto e o evento é o resultado de uma ação, como a ação de receber um tiro ou de acertar o tiro, podendo acionar um

novo procedimento (um novo tiro, por exemplo).

5. Anatomia do Robô

O robô tem o formato de um taque de guerra, este, dividido em 3 componentes:

- Corpo (Body)
  - Responsável pelos movimentos.
    - Mover para frente;
    - Mover para trás;
    - Girar para esquerda;
    - Girar para direita;
    - Parar
- Canhão (Gun)
  - Responsável por realizar os disparos.
  - Gira para esquerda e para a direita
- Radar
  - Localiza os inimigos
  - Girar para esquerda e para a direita,

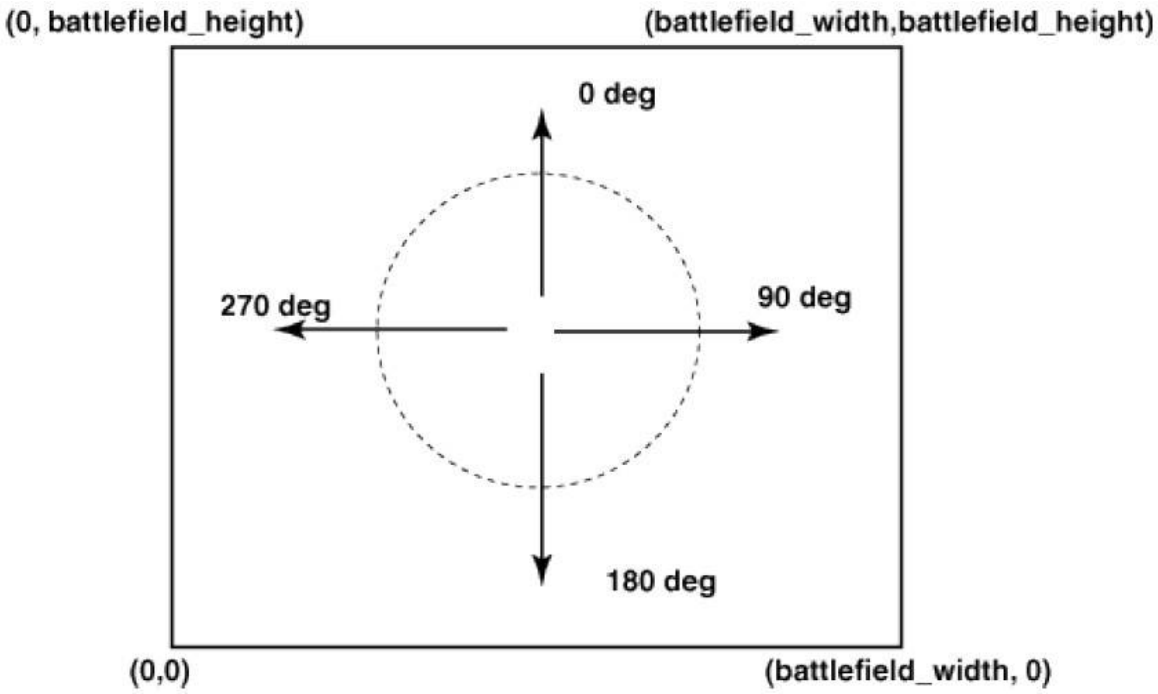


Figura 5. Navegação

O canhão fica por cima do corpo e o radar está montado em cima do canhão. O giro do corpo é acompanhado ou não pelo canhão e o radar.

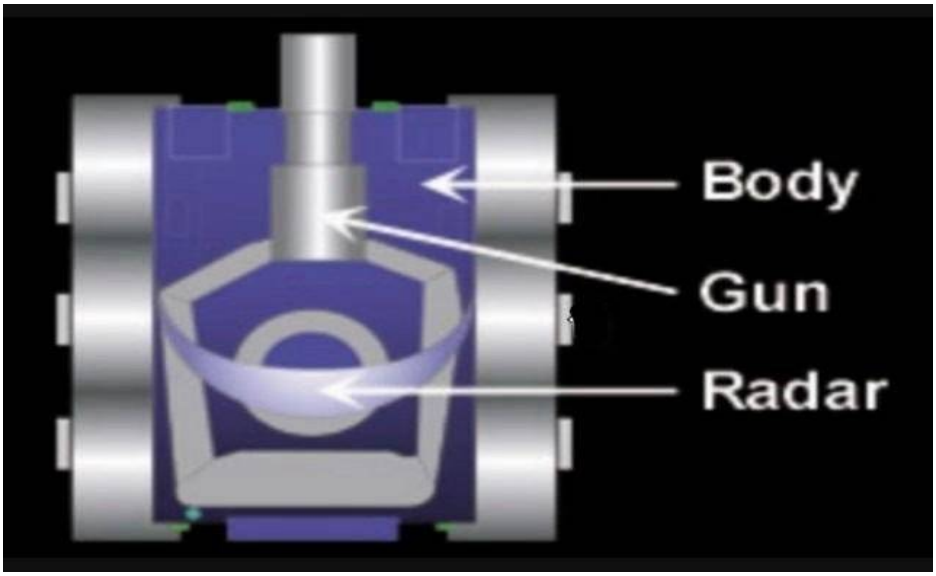


Figura 6. Anatonomia de Robô.

6. Regras

- O jogador deve saber que o robô pode se movimentar para somente para frente, para trás e fazer curvas.
- O canhão (gun) pode virar no sentido horário e anti-horário em 360 graus, e dar tiros(bullet) de força maior que 0 e menor ou igual a 3.
- Se o robô ficar sem energia, ou seja, energia igual a zero, ele fica desabilitado (disabled) e perde os seus movimentos.
- Deve-se saber também que robô é "cego", a única coisa que ele vê são os robôs adversários, capturados pelo radar(radar), o qual não vê os tiros.
- Todos os robôs começam cada round com energia 100, e os que ficarem com energia abaixo de 0 vão sendo eliminados (explodem) restando apenas um, e então começa um novo round.
- No fim de todos os rounds a batalha acaba e aparece uma tabela mostrando a pontuação e a colocação.

Os robôs possuem também uma barra de energia, onde eles podem tanto perder energia como ganhar.

O robô perde energia quando:

- Atingido por um tiro:
  - perde uma quantidade de energia equivalente a 4 vezes a potência do tiro. Se

a potência for maior ou igual a 2, é realizado um dano adicional de 2 vezes a (potência -1).

- Realizando um tiro:
  - Perde a quantidade equivalente à potência do tiro realizado;
- Se bater em outro robô, os dois perdem (energia – 1).

O robô só ganha energia se atingir outro robô com um tiro, onde ganha 3 vezes a potência do tiro realizado.

7. Instalação

Inicialmente, é preciso o Java Runtime Environment (JRE) ou Java Developer Kit (JDK), observe que o JRE não inclui o compilador Java padrão (javac), mas o JDK já o possui.

Instalação do Java

- No windows:
  - Vá para a página http://www.java.com/pt_BR/download/manual.jsp
  - Faça o download da versão mais nova e após o download, execute o instalador.
- No Linux (Ubuntu)
  - Acesse http://java.com e clique no botão Download
  - Vá para o diretório no qual deseja instalar o arquivo. Digite cd <*nome do caminho do diretório*>;
  - Mova o archive binário .tar.gz para o diretório atual.
  - Desempacote a tarball e instale o Java `tar zxvf jre-7u7-linux-i586.tar.gz`
  - Os arquivos Java são instalados no diretório jre1.7.0_07 no diretório atual.

Instalando o Robocode

- Acesse a página http://sourceforge.net/projects/robocode/files/
- Faça o download do arquivo `robocode-1.8.3.0-setup.jar`,
- Siga os passos da instalação e pronto.

8. Campo de batalha

As batalhas são realizadas em um campo de virtual, onde as dimensões desses campos são indicadas pelo jogador, na qual são medidas em pixels. As batalhas podem ser realizadas com diversos tanques simultaneamente, mas toda batalha somente chega ao

fim quando restar um único tanque.

O sistema de pontuação diz quem é o vencedor. Os pontos são atribuídos da seguinte maneira:

- Sobrevivência:
  - Sempre que um tanque é destruído, todos os tanques que ainda estão na batalha recebem 50 pontos;
- Sobrevivente
  - O último tanque restante recebe um bônus de 10 pontos para cada tanque que foi destruído, independente se tais tanques tenha sido destruído pelo sobrevivente ou não;
- Danos por tiro:
  - Cada tanque recebe um ponto para cada disparo acertado.
- Danos por colisão
  - A cada 1 ponto de dano causado em outro robô através de uma colisão, é recebido 2 pontos;

Figura 7. Campo de batalha no Robocode

A posição inicial de cada tanque é aleatória. Frequentemente as batalhas são repetidas algumas vezes para eliminar a possibilidade de determinado tanque ter vantagem devido a um posicionamento inicial favorável.

Caracteristica dos pontos:

- Modo Survival: 50 pontos toda vez que um inimigo morre;
- Modo Survival bonus: 10 vezes o número de inimigos;
- Bullet damage: 1 ponto por ponto de dano no inimigo;
- Bullet bonus: 20% do dano causado a um inimigo se for você quem o disparou;
- Ram damage: 2 pontos por ponto de dano ao inimigo numa colisão;
- Ram bonus: 30% do dano.

9. Exemplo de um Robo

Ao iniciar com o Robocode, o código de um pequeno robô (HelloWorld) é disponibilizado como inicio para sua implementação.

```
package robot;
import robocode.*;

public class Meu extends Robot {

	public void run() {

		while(true) {

			ahead(100);
			turnGunRight(360);
			back(100);
			turnGunRight(360);
		}
	}

	public void onScannedRobot(ScannedRobotEvent e) {
		fire(1);
	}

	public void onHitByBullet(HitByBulletEvent e) {
		turnLeft(90 - e.getBearing());
	}

}
```

10. Alguns Métodos

- Métodos para a movimentação do Robô

Comando	Parâmetro	Descrição
ahead(*double* **)**	a distância que o robô deverá percorrer.	Movimenta o robô para frente, uma distância x dada por parâmetro. Se o robô bater em outro, ou na parede antes de completar a distancia desejada o método é interrompido.
back(*double* **)**	a distância que o robô deverá percorrer.	Semelhante ao método anterior, a única diferença é que o robô move para tras.
turnRight(*double* **)**	o ângulo em graus que o robô deverá girar.	Gira o robô para a direita (sentido horário).
turnLeft(*double* **)**	o ângulo em graus que o robô deverá girar.	Gira o robô para a esquerda (sentido anti-horário).
turnGunRigth(*double* **)**	o ângulo em graus que o canhão deverá girar	Gira o canhão para a direita.
turnGunLeft(*double* **)**	o ângulo em graus que o canhão deverá girar	Gira o canhão para a esquerda.
turnRadarRigth(*double* **)**	o ângulo em graus que o radar deverá girar	Gira o radar para a direita.
turnRadarLeft(*double* **)**	o ângulo em graus que o radar deverá girar	Gira o radar para a esquerda.

- Métodos para dar tiro

Comando	Parâmetro	Descrição
fire(*double* **)**	a força do tiro, e subtraido da energia de seu robô.	Atira imediatamente na força mandada por parâmetro, de 0.1 até 3. Se mandar um tiro maior que 3 ele considera

		força 3.
fireBullet(*double* **)**	a força do tiro, e subtraido da energia de seu robô.	A diferença do método anterior é que ele é uma função e retorna um valor do tipo *Bullet*, além disso, manda outro tiro em seguida, este com mais velocidade, se o primeiro tiro tiver boas possibilidades da acertar.

11. Referências

Larsen, F. N. “ReadMe for Robocode”, http://robocode.sourceforge.net/docs/ReadMe.html

Souza, R. F. “Robocode”, http://www.slideshare.net/rosicleiafrasson/robocode-22009167

Oracle, “O que é Java?”, http://www.java.com/pt_BR/download/whatis_java.jsp

Loadholtes, N. “IBM’s Robocode: A Platform for Learning AI”, http://ai-depot.com/articles/ibms-robocode-a-platform-for-learning-ai/

Utilização de Heurísticas Bio-inspiradas em Sistemas de Inteligência Coletiva para Otimização Combinatória em Redes *Mesh*

Leinylson F. Pereira, Jacks R. N. Fernandes

[1] Departamento de Computação - Universidade Estadual do Piauí (UESPI)
Caixa Postal 64202-220 – Parnaíba – PI – Brazil

leinylson@gmail.com, jacks.renan@hotmail.com

Abstract. *The perspective that have colonies of insects is that there prevails a random movement in search of food, but not quite, there is cooperativity between members, allowing the emergence of a society super-organized and self-adaptive to changes in environment. Aiming at the adoption of collective intelligence and as a way to demonstrate a system with an intelligent behavior, was created the RotÓtima simulator , that making use of mathematical models, provides an environment in which multiple agents representing ants, act cooperatively and indirectly, in finding a solution to the problem of route optimization in mesh networks, demonstrating the efficacy of the technique Bioinspired these simple individuals.*

Resumo. *A perspectiva que se têm das colônias de insetos é que prevalece uma movimentação aleatória em busca de comida, mas não é bem assim, existe uma cooperatividade entre os membros, que permite o surgimento de uma sociedade superorganizada e autoadaptativa às variações do ambiente. Visando a adoção da inteligência coletiva e como forma de demonstração de um sistema dotado de um comportamento inteligente, foi criado o simulador RotÓtima, que fazendo uso de modelos matemáticos, fornece um ambiente no qual múltiplos agentes, representando formigas, atuam cooperativamente e indiretamente, em busca de uma solução para o problema de optimização de rotas em redes de malha, demonstrando a eficácia da técnica bioinspirada nestes simples indivíduos.*

1. Introdução

Uma rede de computadores trata-se de um sistema de comunicação de dados constituído através da interligação de computadores e outros dispositivos, com a finalidade de trocar informações e compartilhar recursos[Azevedo 2005]. A busca por um melhor aproveitamento do tempo e recursos disponíveis durante a transmissão dos dados, levando sempre em consideração as métricas de *Quality of Service*[1] *(QoS)*, vem gerando novos desafios à área de projeto e planejamentos de redes. O interesse em pesquisar o tema deste artigo, justifica-se devido ao aumento do uso da internet[Antonioli 2012]. Em redes de protocolo IP, não existe reserva de recursos para cada usuário, o que torna necessário a minimização do número de saltos entre os nós da rede, traçando rotas físicas que busquem aumentar o desempenho global da rede.

2. Referencial Teórico

2.1. *Wireless Mesh Networks* e Otimização Combinatória

As redes em malha sem fio, ou *Wireless Mesh Networks (WMN's)*, são formadas por dois tipos de dispositivos, os clientes *mesh* e os roteadores *mesh*. Os clientes *mesh* são nós, móveis ou

[1]Qualidade do serviço, isto é, um conjunto de regras que descrevem e determinam a qualidade de um aplicativo/recurso mensurando sua velocidade, erros e a qualidade no envio de dados.

não, que executam as aplicações dos usuários. A comunicação entre esses nós é realizada pelo *backbone*[2][Silva 2011]. O processo de otimização combinatória visa descobrir qual a melhor combinação dos recursos disponíveis e seus atributos para aperfeiçoar seu uso. Conforme o fluxograma da figura 2[Souza 2008], o campo da inteligência coletiva, corresponde à abordagem adotada durante a elaboração desta pesquisa.

Figura 1. Taxonomia da Otimização Combinatória

2.2. Sistemas Inteligentes e Inteligência Coletiva

Qualquer tentativa de projetar algoritmos ou técnicas de resolução distribuída de problemas inspirada pelo comportamento coletivo de insetos sociais e outras sociedades de animais, constituem uma inteligência de enxames[Bonabeau et al. 1999]. Um sistema coletivo é aquele constituído por um grupo de agentes aptos a interagirem entre si e com o ambiente[Castro et al. 2004]. Do mesmo modo como fazem as formigas reais, as formigas artificiais também modificam aspectos do ambiente através do feromônio[3] artificial. Assim como existe a evaporação dos feromônios reais, também nos feromônios artificiais é utilizado um mecanismo que simula tal evaporação[Dorigo 1992], responsável por modificar a informação do feromônio artificial através do tempo.

3. Trabalhos Relacionados e Aplicações

O Algoritmo de Colônia de Formigas (ACF) foi utilizado primeiramente na resolução do problema do caixeiro viajante[Dorigo et al. 1999]. Posteriormente, o modelo foi indicado para solucionar o problema da configuração de redes[Neto 2000]. Contudo, alguns destes trabalhos[Balaprakash et al. 2009] foram executados sem se desprenderem totalmente do problema do caixeiro viajante, que serviu para a proposta original.

Configuração de Redes de Distribuição Via Algoritmo de Formigas: O problema da configuração de redes, seja de distribuição de energia elétrica, abastecimento de água, dentre outras, pode ser visto como um problema de otimização combinatória típico, que consiste em eleger quais ligações ativar para que a rede tenha configuração ótima[Neto 2000].

Agrupamento por Colônias de Formigas: Os algoritmos de clusterização[4] baseados em colônias de formigas são aplicáveis a problemas de análise exploratória de dados, onde

[2]Rede de alta velocidade formada por linhas de comunicação e hardware de transmissão e recepção, a qual conectam-se todos os provedores de acesso à internet. É a espinha dorsal da Internet.

[3]Os feromônios ou as feromonas, são substâncias químicas que quando emitidas/excretadas são capazes de suscitar reações específicas de tipo fisiológico e/ou comportamental em outros membros que estejam num determinado raio do espaço físico ocupado pelo excretor.

[4]Junção, união, agregação, integração. Agrupamento de coisas ou de atividades semelhantes.

um conjunto de dados não rotulados está disponível e alguma informação deve ser extraída (inferida) destes dados[Castro et al. 2004]. O modelo foi aplicado[LVCoN 1999], a números (objetos) mostrados em uma grade (matriz de dados), como resultado, obteve-se o agrupamento de objetos similares.

Adaptação Social do Conhecimento: Utilizada inicialmente para determinar pesos e definir a arquitetura de Redes Neurais Artificiais (RNA), a técnica de Otimização por Enxame de Partículas (*Particle Swarm Optimization (PSO)*) busca simular a habilidade humana de processar conhecimento[5]. O modelo foi aplicado[LVCoN 1999] visando a otimização de funções contínuas de uma e duas variáveis.

4. Simulador RotÓtima

4.1. Modelagem do Ambiente de Simulação

Utilizando-se de interfaces gráficas amigáveis, o ambiente simula o movimento forrageiro das formigas, conforme a figura 2

Figura 2. Interfaces Gráficas do Usuário

4.2. Probabilidade de Transição

O algoritmo pressupõe que a formiga lembra dos locais já visitados, existe uma lista de nós que contém os nós já visitados pela formiga, constituindo uma lista tabu contendo os n últimos elementos visitados[Dorigo et al. 1996]. Estando no nó i, a formiga k escolhe o nó j, com uma probabilidade dada pela equação 1, dentre os nós que ainda não foram visitados.

$$P_{ij}^{k}(t) = \begin{cases} \frac{[\tau_{i,j}(t)]^{\alpha}}{\sum_{j \in N_i}[\tau_{i,j}(t)]^{\alpha}}, & se\ j \in N_i^k \\ 0, & se\ j \notin N_i^k \end{cases} \quad (1)$$

Em que N_i^k representa o conjunto de nós factíveis conectados ao nó i, ainda não visitados pela formiga k, e α é uma constante positiva que determina a inuência relativa da trilha de feromônio e $\tau_{i,j}$ é a quantidade de feromônio sobre a ligação (i, j)

4.3. Qualidade da Solução

A qualidade da solução dada pela equação 2 é expressa como o inverso do comprimento em termos da distância física[6] entre o nó origem e o nó destino[Dorigo 1992].

$$\Delta\tau_{i,j}^{k}(t) = \frac{1}{L^{k}(t)} \quad (2)$$

[5]Cada indivíduo de uma população possui sua própria experiência e é capaz de avaliar a qualidade desta (aprendizagem individua cognitiva) e devido à sociabilidade dos indivíduos, eles possuem conhecimentos sobre o desempenho de seus vizinhos (transmissão cultural).

[6]Qualquer outra medida poderia ser usada, tal como o custo da viagem no caminho ou ainda o número de saltos no caminho, do nó origem ao nó destino.

4.4. Evaporação da Trilha de Feromônio

A equação 3 é utilizada para evitar uma convergência[7] prematura do algoritmo em uma região subótima[Souza 2011].

$$\tau_{i,j}(t) = (1 - \rho).\tau_{i,j}(t) \tag{3}$$

4.5. Atualização da Trilha de Feromônio

Quando uma formiga deposita feromônio, conforme a equação 4, ela está aumentando a probabilidade de que esta conexão seja selecionada por outra formiga.

$$\tau_{i,j}(t+1) = \tau_{i,j}(t) + \sum_{k=1}^{N_f} \Delta\tau_{i,j}^{k}(t) \tag{4}$$

4.6. Critérios de Parada

Critérios de Parada Como critério de parada, equação 5, foram utilizadas as seguintes abordagens: número máximo de iterações excedido, solução aceitável encontrada e estagnação[8].

$$F(x^k(t)) \leq \epsilon \tag{5}$$

5. Análises e Resultados

Nos experimentos, foram retratados os processos de realimentação positiva da trilha de feromônio e realimentação negativa da trilha de feromônio, estigmergia e procura (explotando e explorando). Através de simulações foram obtidas as melhores soluções dentre as rotas exploradas pelos agentes, caracterizando uma solução aceitável para o problema de otimização de roteamento em redes *mesh*, proposto inicialmente.

Para o S-ACO, Dorigo[Dorigo 1992] determinou que cada aresta recebe um pequeno valor x, para indicar a concentração inicial de feromônio $\tau_{i,j}(0)$. Uma quantidade N_f de formigas, são posicionadas no nó origem e para cada iteração do algoritmo, cada formiga incrementalmente (passo a passo) constrói um caminho. Em cada nó, cada formiga executa uma política de decisão para determinar o próximo trecho (*link*) a ser percorrido. Se a formiga k está atualmente localizada no nó i, ela seleciona o próximo nó j, baseada na probabilidade de transição. Depositado pelas formigas, o feromônio variou em virtude dos movimentos de exploração e explotação, obtendo uma solução ótima do grafo, conforme a figura 3 resultantes da simulação realizada em um ambiente com um grafo composto por 12 nós, dos quais o nó '0' representando a origem ou formigueiro e o nó '11' representando o destino ou fonte de alimento, limitada por 3 ciclos, com uma população de 10 formigas, com os valores do feromônio $t = 0.3$, da constante $\alpha = 0.02$ e decaimento do feromônio $\rho = 0.02$. Dentre as soluções encontras pela formiga I, na figura 3a: **[0-1-4-8-6-7-11] = 13**, e pela formiga II na figura 3b: **[0-3-1-4-6-11] = 11**, o melhor percurso foi aquele que possuiu a melhor nota de avaliação da qualidade da solução, obtida pela equação 2, na caso, a formiga II. Na figura 4 tem-se a matriz de feromônio antes do início do processo de forrageamento[9] (à esquerda), e ao término do processo (à direita). A figura 4 (à direita), apresenta a saída gerada pelo algoritmo, com os caminhos mais reforçados de feromônio, constituindo o subótimo global. Ainda que a

[7] Situação na qual permaneceriam forrageando em uma mesma região.

[8] Falta de movimento, atividade, vida; Estado de inércia.

[9] Ação de procurar, remexendo e destroçando.

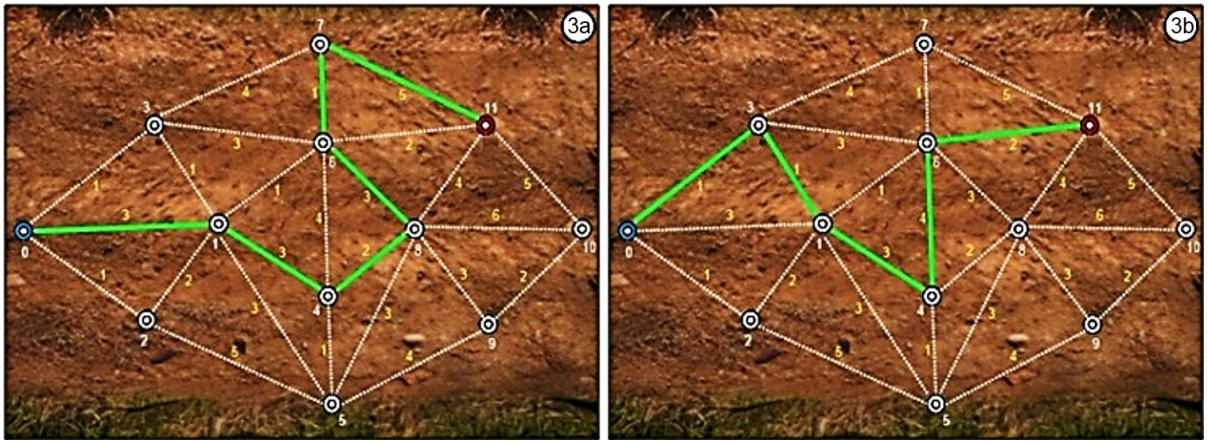


Figura 3. Soluções Encontradas

0.00 0.30 0.30 0.30 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00	0.00 0.35 0.29 0.38 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.30 0.00 0.30 0.30 0.30 0.30 0.30 0.00 0.00 0.00 0.00 0.00	0.35 0.00 0.29 0.38 0.44 0.29 0.29 0.00 0.00 0.00 0.00 0.00
0.30 0.30 0.00 0.00 0.00 0.30 0.00 0.00 0.00 0.00 0.00 0.00	0.29 0.29 0.00 0.00 0.00 0.29 0.00 0.00 0.00 0.00 0.00 0.00
0.30 0.30 0.00 0.00 0.00 0.00 0.30 0.30 0.00 0.00 0.00 0.00	0.38 0.38 0.00 0.00 0.00 0.00 0.29 0.29 0.00 0.00 0.00 0.00
0.00 0.30 0.00 0.00 0.00 0.30 0.30 0.00 0.30 0.00 0.00 0.00	0.00 0.44 0.00 0.00 0.00 0.29 0.38 0.00 0.35 0.00 0.00 0.00
0.00 0.30 0.30 0.00 0.30 0.00 0.00 0.00 0.30 0.30 0.00 0.00	0.00 0.29 0.29 0.00 0.29 0.00 0.00 0.00 0.29 0.29 0.00 0.00
0.00 0.30 0.00 0.30 0.30 0.00 0.00 0.30 0.30 0.00 0.00 0.30	0.00 0.29 0.00 0.29 0.38 0.00 0.00 0.35 0.35 0.00 0.00 0.38
0.00 0.00 0.00 0.30 0.00 0.00 0.30 0.00 0.00 0.00 0.00 0.30	0.00 0.00 0.00 0.29 0.00 0.00 0.35 0.00 0.00 0.00 0.00 0.35
0.00 0.00 0.00 0.00 0.30 0.30 0.30 0.00 0.00 0.30 0.30 0.30	0.00 0.00 0.00 0.00 0.35 0.29 0.35 0.00 0.00 0.29 0.29 0.29
0.00 0.00 0.00 0.00 0.00 0.30 0.00 0.00 0.30 0.00 0.30 0.00	0.00 0.00 0.00 0.00 0.00 0.29 0.00 0.00 0.29 0.00 0.29 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.30 0.30 0.00 0.30	0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.29 0.29 0.00 0.29
0.00 0.00 0.00 0.00 0.00 0.00 0.30 0.30 0.30 0.00 0.30 0.00	0.00 0.00 0.00 0.00 0.00 0.00 0.38 0.35 0.29 0.00 0.29 0.00

Figura 4. Matrizes de Feromônio Inicial (à esquerda) e Final (à direita)

natureza do algoritmo seja estocástica[10], a forte concentração de feromônio nos caminhos por elas visitados força-a ainda mais a realizar continuamente o mesmo trajeto, verifica-se então, um acúmulo considerável das taxas de feromônio.

6. Conclusão e Trabalhos Futuros

A finalidade deste trabalho foi de realizar uma demonstração da forma de atuação de agentes inteligentes baseada na inteligência de enxames, demonstrando assim o potencial dos algoritmos aqui abordados. Existem diversas aplicações que podem ser desenvolvidas baseadas nas teorias aqui expostas, tal como o mapeamento automático de ambientes por máquinas, que necessita de um algoritmo que torne eficaz a realização das tarefas de exploração, construção de grafos a partir do caminho percorrido de forma a orientar a locomoção e reconhecimento do território, quando usado em conjunto com técnicas de visão computacional. Para que se tenham aplicações bem sucedidas, é preciso que as fundamentações teóricas sejam bem estabelecidas.

Além disso, é importante não apenas simular, mas programar em robôs reais as técnicas aqui expostas, constatando-se assim, as restrições de comunicação, processamento, monitoramento, sensoriamento e de desempenho. Pode-se ter em vista uma possível expansão do algoritmo, tal como o uso conjunto de vários tipos de feromônio; navegação de ambientes utilizando robôs voadores *(quadcopters)* ou em pequenos robôs que lembram formigas *(i-swarm)*, para fins de simulação, ou ainda a utilização do algoritmo base, para criação de *frameworks*[11] que façam uso de agentes inteligentes na otimização de rotinas computacionais.

[10]Processos que estão sob controle do acaso, aleatórios, onde o passado não tem vínculo com o futuro.

[11]Conjunto de classes implementadas em uma linguagem específica, usadas para auxiliar o desenvolvimento de software.

7. References

Referências

Antonioli, L. (2012). **Estatísticas, Dados e Projeções Atuais Sobre a Internet no Brasil**. http://tobeguarany.com/internet_no_brasil.php, acesso em: 22 jan. 2013.

Azevedo, N. (2005). **Guia de Estudo: Redes e Internet**. Escola Básica Integrada de Angra do Heroísmo - Tecnologias da Informação e Comunicação (TIC).

Balaprakash, P., Birattari, M., Stützle, T., Yuan, Z., and Dorigo, M. (2009). **Estimation-based Ant Colony Optimization Algorithms for the Probabilistic Travelling Salesman Problem**. Swarm Intelligence, vol. 3, p. 223-242, 3ª ed., issn: 1935-3812, http://iridia.ulb.ac.be/IridiaTrSeries/rev/IridiaTr2008-020r001.pdf, acesso em: 04 jan. 2013.

Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). **Swarm Intelligence: From Natural to Artificial Systems**.

Castro, L., N, and Zuben, J. V. (2004). **Recent Developments in Biologically Inspired Computing**. Idea Group Publishing, ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/ia006_03/tópico4_03.pdf, acesso em: 26 set. 2012.

Dorigo, M. (1992). **Ant Colony Optimization**. http://www.scholarpedia.org/Antcolony, acesso em: 28 nov. 2012.

Dorigo, M., Caro, G. D., and Gambardella, L. M. (1999). **Ant Algorithms For Discrete Optimization**. Artificial Life. vol. 5, p. 97-116.

Dorigo, M., Maniezzo, V., and Colorni, A. (1996). **The Ant System: Optimization by a Colony of Cooperating Agents**. IEEE Transactions on Systems, Man, and Cybernetics - Part B, vol. 26, p. 29-41.

LVCoN (1999). **Laboratório Virtual em Computação Natural**. http://www.lvcon.computacaonatural.com.br, acesso em: 28 dez. 2012.

Neto, J. P. C. (2000). **Configuração de Redes de Distribuição via Algoritmo de Formigas**. http://www.dee.ufcg.edu.br/ pet/jornal/projs/proj6.html, acesso em: 26 dez. 2013.

Silva, H. W. (2011). **Um Esquema de Seleção de Rotas para o Balanceamento de Segurança e Desempenho em Redes em Malha Sem Fio**. Programa de Mestrado em Informática Aplicada , Universidade de Fortaleza - UNIFOR, Ceará, cap. 2, p. 6.

Souza, M. J. F. (2011). **Inteligência Computacional para Otimização**. Departamento de Computação, Instituto de Ciências Exatas e Biológicas, Universidade Federal de Ouro Preto, Ouro Preto, MG.

Souza, R. C. T. (2008). **Heurísticas Bioinspiradas de Otimização Combinatória**. GESTÃO – Revista de Administração e Sistemas de Informação, vol. 10, nº 10, http://www.faculdadeexpoente.edu.br/upload/noticiasarquivos/1243985982.PDF, acesso em: 30 jan. 2013.

Uma proposta para classificação de rotas em Redes de Sensores sem Fio baseada em sistemas fuzzy e otimização por colônia de formigas

José Victor Vasconcelos Sobral[1], Aldir Silva Sousa[2], Ricardo A. L. Rabêlo[1]

[1]Universidade Federal do Piauí (UFPI)
Programa de Pós-Graduação em Ciência da Computação
Teresina, PI – Brasil

[2]Centro de Ensino Unificado de Teresina (CEUT)
Laboratório de Ciência da Computação
Teresina, PI – Brazil

jose.v.sobral@ieee.org, aldirss@yahoo.com.br, ricardor_usp@ieee.org

Abstract. *The Wireless Sensors Networks (WSNs) are composed of small sensors nodes capable of sensing and transmitting data related to some phenomenon in the environment. The sensors nodes have severe constraints, such as lower processing and storage capacity, and, generally, are operated only by a limited-energy battery. The WSN Routing protocols must have auto configuration features in order to find out which is the best route for communication, thus increasing delivery assurance and decreasing the energy consumption between nodes that comprise the network. This paper presents a proposal for routes classification using Fuzzy Inference Systems to assist a routing protocol in a WSN. The Fuzzy System is used to estimate the degree of the route quality, based on the number of hops and the lowest energy level among the nodes that form the route. An Ant Colony Optimization (ACO) algorithm is used to adjust the rule base of the fuzzy system in order to improve the classification strategy of route, hence increasing the energy efficiency of the network. The simulations indicated that the proposal is effective from the point of view of the energy, the number of received messages, the necessary time to send a specified number of messages and the cost of received messages when compared with other approaches.*

Resumo. *As redes de sensores sem fio (RSSF) são compostas por pequenos sensores capazes de sentir e transmitir dados de alguns fenômenos do ambiente em que estão inseridos. Os nós sensores possuem restrições severas no que diz respeito a poder de processamento e capacidade de armazenamento, além de, geralmente, operarem com uma energia limitada (baterias). Os protocolos de roteamento das RSSF devem possuir características de autoconfiguração, afim de obter o melhor caminho para comunicação entre os nós, aumentado, assim, a garantia de entrega dos dados e a diminuição do consumo de energia pelos nós da rede. Este trabalho apresenta uma proposta para classificação de rotas usando um sistema de inferência fuzzy para auxiliar os protocolos de roteamento das RSSF. O sistema de inferência fuzzy estima o valor da qualidade da rota baseado no número de saltos e no menor nível de energia dos nós que compõem a rota. Um algoritmo de otimização por colônia de formigas (ACO)*

é utilizado para ajustar a base de regras do sistema de inferência fuzzy, a fim de melhorar a estratégia de classificação de rotas, para aumentar a eficiência da energia na rede. As simulações indicam que a proposta é eficiente no que diz respeito a redução do consumo de energia, número de mensagens recebidas, tempo para enviar um determinado número de mensagens e custo médio por mensagem recebida, quando comparado a outras propostas.

1. Introdução

Uma rede de sensores sem fio pode ser definida como um conjunto de nós sensores que têm a função de captar as ações do meio em que estão inseridos e, por meio de comunicação sem fio, podem transmitir informações para outros nós da rede [Akyildiz et al. 2002]. As redes de sensores possuem limitações que a tornam bastante diferentes das redes tradicionais. As principais restrições das redes de sensores sem fio (RSSF) estão ligadas ao baixo poder de processamento e memória, e a uma quantidade limitada de energia. As RSSF geralmente possuem nós sensores e *sink nodes*. Os nós sensores são responsáveis por extrair informações do meio em que estão inseridos, enquanto os *sink nodes* são responsáveis por receber informações dos nós sensores e repassá-las a um observador.

Segundo [Akyildiz et al. 2002], as principais áreas de aplicação de RSSF são: aplicações militares, aplicações ambientais, aplicações na área de saúde, aplicações domésticas, áreas de produção industrial e de distribuição de energia, água e gás.

Na maioria das vezes, as RSSF são implantadas em locais de difícil acesso ou que não permitem a presença humana constantemente [Mainwaring et al. 2002]. Com isso, torna-se inviável a substituição de baterias ou nós inteiros após o término do seu tempo de vida. Para aumentar o tempo de vida de uma RSSF, é necessário reduzir a utilização de bateria, ou seja, diminuir o número de processamento e transmissões de rádio nos nós sensores. É extremamente importante diminuir a utilização do rádio pelos nós sensores se se pretende diminuir os gastos de energia. Com isso, dá-se a importância de um roteamento eficiente.

A proposta apresentada neste trabalho tem como objetivo utilizar um sistema de inferência *fuzzy* [Zadeh 1975] posteriormente otimizado por um algoritmo de colônia de formigas [Dorigo et al. 1996] para avaliar as rotas criadas pelo protocolo de roteamento e auxiliá-lo na escolha da melhor rota em um determinado momento [Rabelo et al. 2013][Sobral et al. 2013]. Para isto, o sistema *fuzzy* deverá receber como entrada do processo de inferência, o número de saltos da rota e o menor nível de energia dos nós da rota e apresentar como resultado a qualidade da rota baseando-se nos dados de entrada. O protocolo de roteamento escolhido para realização dos estudos é o *Directed Diffusion* [Intanagonwiwat et al. 2003]. Este protocolo foi escolhido por ser um dos protocolos *multipath-based* mais referenciados na literatura.

A seção a seguir faz uma breve apresentação sobre protocolos de roteamento em RSSF. Na seção 3 é apresentada a abordagem utilizada neste trabalho. A seção 4 expõe os resultados obtidos através das simulações realizadas. A ultima seção apresenta as conclusões do trabalho.

2. Protocolos de Roteamento em RSSF

O protocolo de roteamento, em RSSF, é responsável por determinar por onde os dados captados pelos nós sensores devem passar até chegar ao *sink node* [Al-Karaki and Kamal 2004]. Os protocolos de roteamento são classificados, quanto ao seu modo de operação, em: *multipath-based*, *query-based*, *negotiation-based*, *QoS-based* e *coherent-based*. A proposta descrita neste trabalho é capaz de adaptar-se a qualquer protocolo de roteamento *multipath-based*. Estes são protocolos em que são criadas várias rotas para o *sink node*, a fim de aumentar a tolerância a falha. Para verificação da abordagem proposta nesse trabalho, o sistema classificador de rotas foi utilizado junto ao protocolo *Directed Diffusion*, que é um protocolo *multipath-based*. Todo o funcionamento do protocolo é descrito em [Intanagonwiwat et al. 2003].

3. Descrição da Protosta

Segundo [Kulkarni et al. 2011], diversas técnicas de inteligência computacional (IC) têm sido utilizadas para resolver os problemas das RSSF. Os paradigmas de IC vêm sendo utilizados com sucesso para solucionar desafios tais como agregação e fusão de dados, roteamento ciente de energia, agendamento de tarefas, segurança e cobertura e conectividade.

A proposta deste trabalho consiste em utilizar técnicas de IC para a solução dos problemas de roteamento existente nas RSSF. Para tal, faz-se uso de um sistema de inferência *fuzzy* classificador de rotas que, junto a um protocolo de roteamento, pode ser capaz de aumentar o tempo de vida da rede e a qualidade dos serviços prestados por esta. As subseções seguintes mostram como funciona o sistema de inferência classificador de rotas e posteriormente como ocorre a otimização da base de regras do sistema de inferência *fuzzy* através do algoritmo de colônia de formigas (ACO).

3.1. Sistema de Inferência *Fuzzy* Classificador de Rotas

A proposta descrita neste trabalho, como supracitado, tem como objetivo desenvolver um sistema de inferência *fuzzy* capaz de auxiliar os protocolos de roteamento no momento da seleção de rotas.

A seleção das rotas ocorre no momento em que o protocolo de roteamento precisa selecionar uma rota, dentre as várias existentes, para ser reforçada. O reforço da rota serve para que todos os dados da rede trafeguem apenas por um caminho, evitando o envio de informações redundantes e diminuindo o número de transmissões da rede, o que impacta na redução do consumo de energia da mesma. Assim, dá-se a grande importância de uma seleção ótima da rota a ser reforçada.

Para classificação das rotas, o sistema de inferência *fuzzy* utiliza as informações do menor nível de energia dos nós que compõem a rota e o número de saltos que a rota possui deste a sua origem ao *sink node*. Estas informações chegam ao sistema classificador de rotas, através de mensagens de controle que trafegam pela rede. Ao receber estas mensagens controle, o *sink node*, utilizando o sistema proposto neste trabalho, efetua a classificação das rotas para uma posterior seleção da rota ótima.

Após a realização de alguns testes no sistema classificador de rotas, observou-se que era necessário uma otimização na base de regras do sistema de *inferência fuzzy* para

que as classificações pudessem ser mais precisas. A sessão a seguir descreve como ocorre o processo de otimização da base de regras do sistema fuzzy através da utilização de algoritmos de otimização por colônia de formigas.

3.2. Algoritmo ACO Ajustando a Base de Dados do Sistema de Inferência *Fuzzy*

Na proposta de otimização da abordagem apresentada neste trabalho, um algoritmo de otimização por colônia de formigas foi usado para ajustar de forma otimizada a base de regras do sistema de inferência *fuzzy*. Desta forma, o caminho a ser percorrido por uma formiga artificial é considerado como uma combinação de termos primários para a variável linguística de saída (qualidade da rota) para todas as regras da base de regras.

Portanto, para cada regra, os valores N_{pt} linguísticos estão disponíveis para ser selecionado, onde N_{pt} é o número de termos primários para a variável linguística de saída. Durante o caminho, a formiga tem de escolher um termo primário para cada regra de um total de opções N_{pt}. Desta forma, a especificação completa da base de regra do sistema de inferência *fuzzy* é toda dada pelo caminho de uma formiga.

Supondo-se que N_r seja o número de regras linguísticas presentes na base de regras, existindo $N_r^{N_{pt}}$ combinações associadas a variável linguística de saída. Como a base de regra diz respeito ao mapeamento de valores de entrada para o valor de saída, um ajuste ideal da base de regra aumenta os resultados produzidos pelo sistema de inferência *fuzzy*. Para o propósito deste trabalho, o resultado produzido pelo sistema de inferência *fuzzy* é o grau de qualidade das rotas (qualidade rota).

Quanto melhor o resultado produzido pelo sistema *fuzzy*, maior é o tempo de vida da RSSF. A meta do trabalho é encontrar uma boa combinação que maximiza o desempenho do sistema de inferência *fuzzy* para classificar as rotas em RSSF. Portanto, após a fase de treinamento (aprendizagem) via algoritmo ACO, o sistema de inferência *fuzzy* é perfeitamente ajustado e está pronto para ser incorporado em um *sink node* de uma Rede de Sensores Sem Fio real, para classificar as rotas associadas a si mesmo. A qualidade da rota é utilizada pelo protocolo de roteamento para selecionar um caminho especifico para enviar uma mensagem.

O sistema de inferência *fuzzy* proposto possui duas variáveis de entrada: o nível de energia mais baixo dentre os nós que compõem a rota e o número de saltos necessários para o envio da mensagem para o *sink node*. A definição dos valores *fuzzy* para cada variável de entrada foi feita previamente, com base nos conhecimentos de especialistas. Cinco termos primários foram definidos para a variável relacionada com o nível de energia, e três termos primários foram definidos para a variável associada ao número de saltos. Dessa forma, a base de regras contém quinze regras. A variável de saída, que determina o grau de qualidade da rota tem cinco termos primários.

Portanto, para cada regra, cinco opções estão disponíveis para o valor linguístico. Dentre as 15^5 (759375) combinações, as formigas artificiais têm que encontrar uma configuração boa para as regras linguísticas. Além do uso de feromônio artificial para ajudar na escolha de um caminho especificado pelas formigas, o algoritmo *Ant System* incorpora uma função heurística. A inclusão de uma informação heurística, normalmente, resulta em melhores soluções, mas requer informações especializadas relacionadas ao problema a ser resolvido. O problema de designar a informação heurística é resolvido,

usando o conhecimento prévio do especialista. Portanto, a experiência acumulada dos especialistas é utilizada para ajudar o processo de tomada de decisão das formigas.

A seção a seguir expõe os resultados obtidos pela utilização do protocolo *Directed Diffusion* com o auxilio do sistema de inferência *fuzzy* classificador de rotas ajustado pelo algoritmo ACO, comparando-o ao protocolo com o sistema classificador de rotas ajustado por um especialista e ao protocolo de roteamento Directed Diffusion puro.

4. Resultados

Os resultados exibem a comparação de três cenários, no primeiro cenário foi utilizado o protocolo de roteamento *Directed Diffusion* puro (DD). No segundo cenário foi utilizado o protocolo *Direcered Diffusion* juntamente com o sistema *fuzzy* classificador de rotas com uma base de regras definidas por um especialista (DD-Fuzzy). Na terceira abordagem, foi utilizado o protocolo *Directed Diffusion* juntamente com o sistema classificador de rotas *fuzzy* com a base de dados ajustada pelo algoritmo de colônia de formigas (DD-ACO-Fuzzy). Para a realização das simulações foi utilizado o simulador Sinalgo.

As métricas utilizadas para comparar os cenários forram: número de mensagens recebidas, energia residual da rede, número de mensagens versus tempo de simulação e custo por mensagem recebida. A métrica do número de mensagens recebidas é referente ao número de mensagens que o *sink node* recebeu até um determinado momento da simulação. A métrica de energia residual se refere ao somatório das energias restantes em cada nó da rede em um determinado momento. A métrica de número de mensagens *versus* tempo de simulação avalia quanto tempo um cenário levou para enviar um determinado número de mensagens. A métrica de custo por mensagem recebida exibe o custo médio para cada mensagem recebida pelo *sink node*, e este custo é calculado com base no número de mensagens recebidas pelo *sink node* e pela energia residual em um determinado momento. Vale ressaltar que a unidade de medida de tempo do simulador é chamada de *round*.

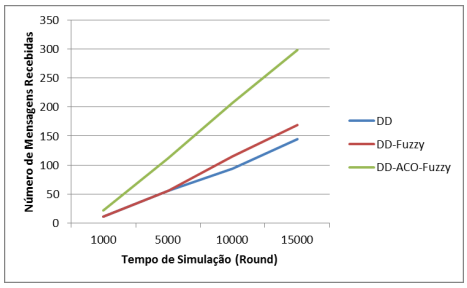


Figura 1. Gráfico de Número de Mensagens Recebidas x Tempo de Simulação

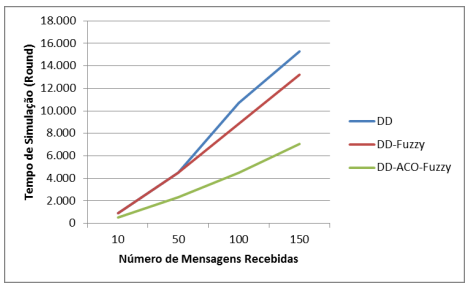


Figura 2. Gráfico de Tempo de Simulação x Número de Mensagens Recebidas

A Figura 1 exibe os resultados obtidos para a métrica de número de mensagens recebidas comparando os três cenários supra descritos. O cenário DD obteve resultados bem modestos, conseguindo enviar cerca de 150 mensagens em 15000 rounds (unidade de tempo do simulador), o cenário DD-Fuzzy conseguiu ser um pouco melhor enviando cerca de 160 mensagens nos mesmos 15000 rounds, já a abordagem descrita neste trabalho, DD-ACO-Fuzzy, superou os demais cenários, enviando cerca de 300 mensagens nos mesmos 15000 rounds.

A Figura 2 complementa a Figura 1, pois nela são expostos os resultados da métrica de número de mensagens versus tempo de simulação. Para enviar 150 mensagens o cenário DD gastou cerca de 15000 rounds, enquanto o cenário DD-Fuzzy gastou cerca de 13000 rounds. Novamente, a proposta descrita neste trabalho mostra-se superior às demais ao conseguir enviar as mesmas 150 mensagens em menos de 8000 rounds.

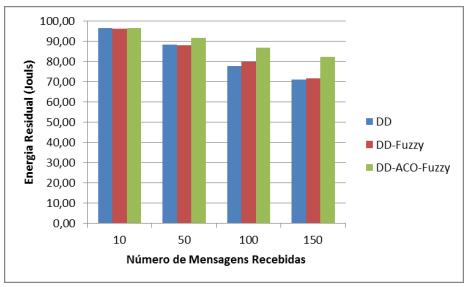


Figura 3. Gráfico de Energia Residual x Número de Mensagens Recebidas

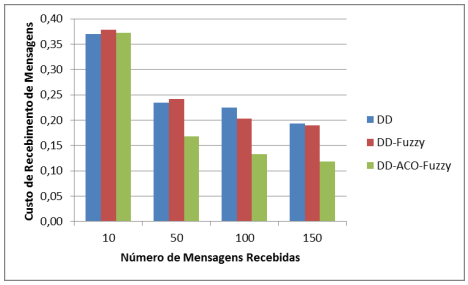


Figura 4. Gráfico de Custo por Recebimento de Mensagem x Número de Mensagens Recebidas

A métrica de energia residual da rede é apresentada na Figura 3. No gráfico, observa-se que, no cenário DD, após o sink node receber 150 mensagens a rede já consumiu cerca 70% da sua capacidade de energia. O cenário DD-Fuzzy, após as 150 mensagens, a energia residual era, aproximadamente, 71%, enquanto no cenário DD-ACO-Fuzzy, para as mesmas 150 mensagens, a energia residual da rede era maior do que 80%. Figura

A Figura 4 expõe os resultados para a métrica de custo por mensagem recebida. Novamente, a proposta descrita neste trabalho consegue superar as demais. O cenário DD-ACO-Fuzzy consegue ter um custo por mensagens recebida, aproximadamente, 60% inferior aos demais cenários.

A sessão seguinte apresenta a conclusão do trabalho baseado nos resultados obtidos pelas simulações.

5. Conclusão

Os resultados expostos mostram que o cenário *Directed Diffussion* com *fuzzy* ajustado pelo ACO, para todas as métricas, é melhor que todos os outros cenários, mostrando resultados positivos em relação ao número de mensagens recebidas, energia residual e custo por mensagem recebida. Portanto, a inclusão de um sistema de inferência *fuzzy* é capaz de melhorar o uso dos limitados recursos computacionais associados às redes de sensores sem fio. Embora o uso de um sistema de inferência *fuzzy* ajustado de forma errada possa fazer bom uso das informações para classificação de rotas, este tipo de ajustado não é melhor do que ajuste automático que o ajuste automático pelo ACO. O algoritmo ACO é capaz de explorar o espaço de busca e identificar as melhores regiões a serem exploradas, deste modo pode-se otimizar os benefícios da utilização de um sistema de inferência *fuzzy* para ajudar um protocolo de roteamento.

Referências

Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002). Wireless sensor networks: a survey. *Computer networks*, 38(4):393–422.

Al-Karaki, J. N. and Kamal, A. E. (2004). Routing techniques in wireless sensor networks: a survey. *Wireless Communications, IEEE*, 11(6):6–28.

Dorigo, M., Maniezzo, V., and Colorni, A. (1996). Ant system: optimization by a colony of cooperating agents. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 26(1):29–41.

Intanagonwiwat, C., Govindan, R., Estrin, D., Heidemann, J., and Silva, F. (2003). Directed diffusion for wireless sensor networking. *Networking, IEEE/ACM Transactions on*, 11(1):2–16.

Kulkarni, R. V., Forster, A., and Venayagamoorthy, G. K. (2011). Computational intelligence in wireless sensor networks: A survey. *Communications Surveys & Tutorials, IEEE*, 13(1):68–96.

Mainwaring, A., Culler, D., Polastre, J., Szewczyk, R., and Anderson, J. (2002). Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 88–97. ACM.

Rabelo, R. A., Sobral, J. V., Araujo, H. S., Baluz, R. A., and Holanda Filho, R. (2013). An approach based on fuzzy inference system and ant colony optimization for improving the performance of routing protocols in wireless sensor networks. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 3244–3251. IEEE.

Sobral, J., Sousa, A., Araujo, H., Baluz, R., Lemos, M., Rabelo, R., et al. (2013). A fuzzy inference system for increasing of survivability and efficiency in wireless sensor networks. In *ICN 2013, The Twelfth International Conference on Networks*, pages 34–41.

Zadeh, L. (1975). The concept of a linguistic variable and its application to approximate reasoning. *Information Sciences*, 8:199–249.

Uma Proposta de Verificação Formal em um Processo de Desenvolvimento Orientado pela UML.

Luciano Kelvin, Thiago de Sousa

Universidade Estadual do Piauí (UESPI) – Parnaíba, PI – Brasil

luciano_kelvin@hotmail.com, thiago@uespi.com

Abstract. *With the advance of modeling tools, formal methods are getting more attention from industrial software engineers. However, it is still hard to convince developers to adopt it because they are not used to mathematical models. On the other hand, UML has become the de facto standard for software modeling since it provides an easy graphical notation and nowadays it is supported by many practical process. In this paper we propose the inclusion of Event-B, a formal language, into ICONIX, a UML-based development process, in order to provide a formal verification mechanism to a software development methodology. So far, we have defined the steps of the proposed process and the translation of the artifacts presented in the first phase of ICONIX to Event-B.*

Resumo. *Com o avanço das ferramentas de modelagem, os métodos formais estão chamando cada vez mais a atenção dos engenheiros de softwares industriais. Porém ainda é difícil convencer os desenvolvedores a adotar esta prática, pois não estão acostumados à modelos matemáticos. Por outro lado, a UML tornou-se padrão para a modelagem de software, fornecendo uma notação gráfica fácil, e atualmente é suportado por diversos processos. Neste trabalho, propomos a inclusão do Event-B, uma linguagem formal, dentro do ICONIX, um processo de desenvolvimento baseado na UML, a fim de proporcionar um mecanismo de verificação formal para esta metodologia de desenvolvimento. Até o momento, nós definimos as etapas do processo proposto, e a tradução dos artefatos apresentados na primeira fase do ICONIX para o Event-B.*

1. Introdução

Métodos formais de verificação estão lentamente ficando sob o foco dos desenvolvedores de sistemas industriais, geralmente por meio de parcerias acadêmicas ou trabalhos colaborativos dentro de projetos de pesquisa financiados pelo governo. Com uma ferramenta amadurecida, os métodos formais tornam-se uma maneira altamente eficiente de especificação formal, modelagem e verificação automática.

Contudo, apesar da eficácia das técnicas, os profissionais da indústria ainda relutam em adotar a abordagem do desenvolvimento formal, que é bastante popular no meio acadêmico. Um dos argumentos utilizados por estes profissionais, é a falta de

escalabilidade da abordagem formal, e também a alta demanda por especialização. Por outro lado, a disposição de usar métodos formais se estes estiverem integrados com uma notação semi-formal bem estabelecida e de fácil entendimento como a UML (Unified Modelling Language) é expressa por muitas empresas. Assim, para muitos profissionais da indústria seria muito útil ter um processo baseado em UML difuso com suporte para técnicas formais.

Neste trabalho nós apresentaremos uma abordagem para a integração do Event-B[1] com ICONIX[2], a fim de proporcionar um processo de desenvolvimento prático e formal baseado na UML. Mais precisamente mostramos como as etapas do ICONIX podem incorporar o formalismo do Event-B a fim de proporcionar técnicas de verificação. Na próxima seção apresentaremos a linguagem Event-B e seus principais conceitos. Na seção 3, mostramos as etapas do ICONIX. Na seção 4, explicamos nossa abordagem, em mais detalhes, mostrando a visão geral da arquitetura, e as principais tarefas de cada fase. Na seção 5 apresentamos as regras de transformação dos dois diagramas (Diagrama de Classe e Caso de Uso) apresentados na primeira fase do ICONIX. Na seção 6, apresentamos alguns trabalhos relacionados, e na seção 7, mostramos alguns trabalhos em curso.

2. Event-B

Event-B é um método formal baseado em modelo de estados, utilizado para modelar sistemas, com base na lógica de predicados e em teoria de conjuntos, onde o mecanismo de refinamento, que nos permite construir um modelo de forma gradual, tornando-o cada vez mais preciso (de um modelo abstrato para um modelo concreto), e checar as inconsistências através da verificação da validade das propriedades do sistema, que podem ser garantidas por obrigações de provas matemáticas. Estas características têm sido apoiadas por uma plataforma *open-source* (baseada na Eclipse IDE) chamada Rodin[3], que é constantemente melhorada pela comunidade através de plugins.

A linguagem Event-B tem dois componentes: contexto e máquina. As máquinas e contextos podem possuir várias relações: a máquina refina outra, enquanto um contexto extende outro. Além disso, uma máquina pode ver vários contextos. Um contexto é utilizado para especificações estáticas, incluindo conjuntos globais, constantes e axiomas, sua estrutura é “Context C1 extends C2 sets S Contants C axioms A end”, onde “C1” e “C2” são os nomes dos contextos, “S” é um conjunto, “C” representa uma constante e “A” representa um axioma. A máquina é aplicada para especificações dinâmicas, contendo variáveis, invariantes e eventos, sua estrutura é “Machine M1 refines M2 sees C1 variables V invariants I events E end”, onde “M1” e “M2” são os nomes das máquinas, “C1” é um contexto visualizado pela máquina “M1”, e “V”, “I” e “E” são as variáveis, invariantes e eventos respectivamente.

Um evento é composto por dois elementos: guardas, que são condições que precisam ser satisfeitas para execução do evento, e as ações que denotam como as variáveis mudam.

3. ICONIX

O processo ICONIX pode ser considerado puro, leve e prático, além de ser uma metodologia extremamente poderosa. Iconix não é tão burocrático como o RUP (Rational Unified Process), o que significa que não gera uma grande quantidade de

documentação. Além disso, o ICONIX usa apenas quatro diagramas (Caso de Uso, Robustez, Sequencia e Classe) e traz ao desenvolvedor para uma "obrigatória" tarefa de verificação entre suas fases.

O processo ICONIX é composto por quatro etapas principais: análise de requisitos, em que o modelo de domínio é produzido, e alguns casos de uso são identificados, a análise do projeto preliminar, onde a análise de robustez[4] é efetuada, a fim de encontrar possíveis erros no caso de uso, e o modelo de domínio é atualizado, após isso é realizado o detalhamento do projeto, onde o diagrama de classe é produzido a partir do domínio, e o diagrama de sequencia é criado a partir do diagrama de robustez; e por último é feita a implementação onde o diagrama de classe e de sequencia são utilizados para guiar a geração do código.

4. Event-B + ICONIX

Podemos ver na figura 1, como a nossa proposta utiliza a anotação do processo ICONIX com as invariantes do Event-B, a fim de proporcionar um mecanismo de verificação formal.

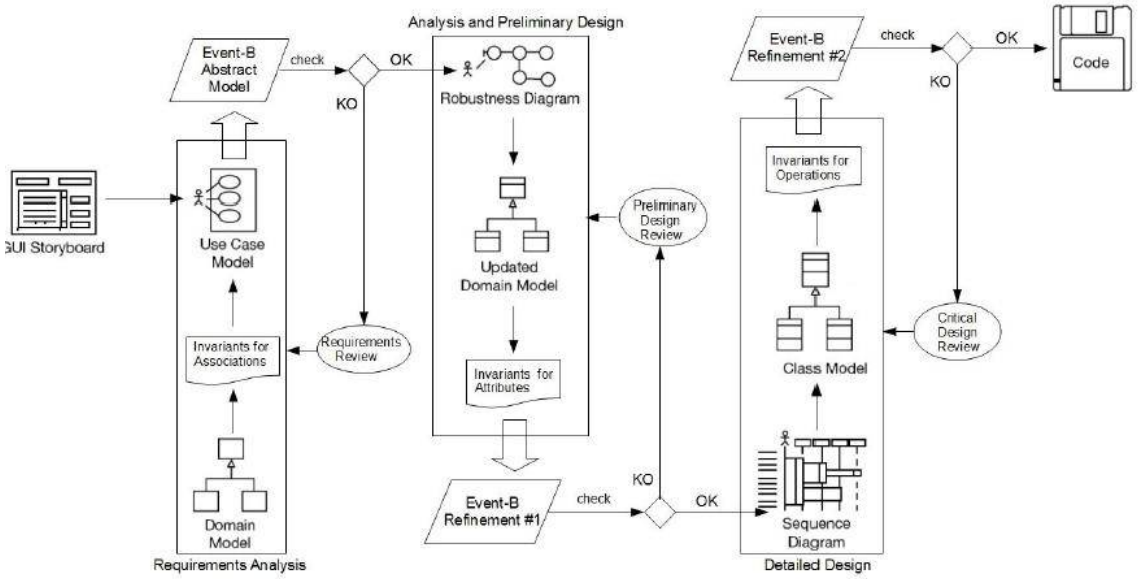


Figura 1. Visão geral da nossa proposta.

A primeira fase se inicia com a concepção do modelo de domínio, Depois disso as invariantes (representadas diretamente na notação Event-B) para as associações do modelo de domínio são refinadas. A primeira etapa termina com a produção do modelo de caso de uso. Por fim estes produtos são convertidos automaticamente para a linguagem Event-B, a fim de compor um modelo abstrato Event-B. Após isto o desenvolvedor pode utilizar a plataforma Rodin para realizar a verificação formal do modelo. Se houver uma violação de qualquer prova de obrigação que foi gerada durante a transformação, o desenvolvedor deve realizar a revisão dos requisitos e corrigir os artefatos. Se não houver nenhum problema detectado o desenvolvedor pode sequir para a próxima etapa.

A segunda etapa inicia-se com a análise de robustez. Após a criação de todo o diagrama de robustez, novas classes e atributos são descobertos, e são utilizados para atualizar o modelo de domínio. Depois disso, as invariantes relacionadas aos novos atributos e

classes são atualizadas e também são atualizados no modelo de domínio. Por fim estes artefatos são traduzidos para o Event-B, a fim de compor o primeiro refinamento do modelo Event-B. Assim o desenvolvedor pode novamente executar as regras dentro da plataforma Rodin, para verificar as inconsistências entre os modelos. Se não houver erros a etapa seguinte pode ser iniciada.

A terceira etapa começa com o desenho do diagrama de sequencia que são derivados aprtir do diagrama de robustez. Após isto o modelo de domínio é atualizado e refinado com os métodos que foram atribuídos as classes. As invariantes assim como as guardas(pré-condições) e ações para cada método das classes são definidos, e finalmente são traduzidos para o Event-B, compondo mais um refinamento do modelo.

Após isto, pode-se novamente executar as regras no Rodin, e verificar a consistência entre os artefatos, e a exatidão do refinamento. Se não for detectado nenhum erro a implementação pode ser iniciada.

5. Modelo de Domínio e Caso de Uso para Event-B

Neste projeto até agora nos concentramos em dois artefatos apresentados na primeira fase do ICONIX: o modelo de domínio e o caso de uso. A tradução destes artefatos para o Event-B já foi implementada utilizando regras QVT(Query/View/Transformation), que foram implementadas dentro da plataforma Rodin a fim de facilitar a integração com a linguagem Event-B com o editor UML.

É importante ressaltar que nossa tradução do modelo de domínio é inspirada no plugin UML-B[5]. É essencial destacar algumas características relacionadas ao ICONIX: Para projetar o modelo de domínio apenas agregação e generalização podem ser usadas como associações. E no Caso de Uso não há nenhuma relação de generalização.

Em nossa abordagem a tradução do modelo de domínio para o Event-B, as classes geram um conjunto global, uma variável, um tipo de invariante e um evento construtor e um evento destrutor. Um atributo gera uma invariante relacionada aquele atributo a uma classe. A generalização é mapeada para uma invariante que faz um subconjunto da classe relacionada. E a agregação produz uma relação entre as classes ligadas variando de acordo com a cardinalidade da relação. Essas regras de transformação são apresentadas na figura 2.

Para a tradução do diagrama de caso de uso para o Event-B cada caso de uso gera um conjunto global, constantes e um axioma. Um ator tem basicamente os mesmo mapeamentos: eles são traduzidos para um evento com uma variável de controle de status. A “precedes” é mapeado como uma variável que é utilizada como guarda em um outro evento. Um “invokes” é transformado em um evento, que é controlado pelos casos de uso relacionados. Estas regras de mapeamento estão apresentadas na figura 3.

Domain Model	Event-B	Example
Diagram	Context	Context School end
Class	SET	SETS Person_SET
	Variable	VARIABLES Person
	Invariant	INVARIANTS inv1: Person ∈ P (Person_SET)
	Constructor Event	Event Cons_Person any self where grd1: self ∈ Person_SET \ Person then act1 : Person := Person ∪ {self} end
	Destructor Event	Event Des_Person any self where grd1: self ∈ Person then act1 : Person := Person \ {self} end
Attribute	Invariant	INVARIANTS inv1: active ∈ Person → BOOL
Generalization	Invariant	INVARIANTS inv1: Student ∈ P (Person)
Aggregation	Invariant	Depends on the multiplicity relation. For a 0..n and 0..n : INVARIANTS inv1: have ∈ SchoolSupplies ↔ SchoolSupplies

Figura 2. Regras de transformação do modelo de domínio para o Event-B. Fonte o autor.

Use Case Diagram	Event-B	Example
Diagram	Sets	Status
	Constants	nostatus, started, ended
	Axiom	axm1: partition{Status, {nostatus}, {started}, {ended}}
Use Case	SET	StudentRegister_SET
	Constants	waitingforStudentRegister
	Variable	control_StudentRegister
	Invariant	inv1 : control_StudentRegister ∈ Status
	INITIALIZATION	act1: control_StudentRegister := nostatus
	Event	EVENT StudentRegister ≙ WHEN grd1: control_StudentRegister = started THEN act3: control_StudentRegister := ended END
	Axiom	axm1: partition{Status, {nostatus}, {started}, {ended}, {waitingforStudentRegister}}
Actor	Variable	control_User
	Invariant	control_User ∈ Status
	INITIALIZATION	act2: control_User := ended
	Event	EVENT Userstarts ≙ WHEN grd2: control_User = ended THEN act3: control_User := started END EVENT User ≙ WHEN grd3: control_User = started THEN Act4: control_User := ended Act5: control_StudentRegister := nostatus END

Use Case Diagram	Event-B	Example
Precedes	Variable	Login_precedes
	Invariant	Inv2: Login_precedes ∈ Status
	INITIALIZATION	act6: Login_precedes := nostatus
	Event	EVENT Login ≙ WHEN grd5 : Login_precedes = nostatus THEN act4 : Login_precedes := ended END EVENT ManageStudentinvokesStudentRegister ≙ WHEN grd6 : Login_precedes = ended THEN END
Invokes	Event	EVENT StudentManagerinvokesStudentRegister ≙ WHEN grd2 : control_StudentManager = started ⋁ (control_StudentManager = waitingforLogin ⋀ control_Login = ended) THEN END EVENT StudentManagerinvokesLogin ≙ WHEN grd3 : control_StudentManager = started ⋁ (control_StudentManager = waitingforStudentRegister ⋀ control_StudentRegister = ended) THEN END EVENT StudentManager ≙ WHEN grd4 : control_StudentManager = started ⋁ (control_StudentManager = waitingforLogin ⋀ control_Login = ended) ⋁ (control_StudentManager = waitingforStudentRegister ⋀ control_StudentRegister = ended) THEN END

Figura 3. Regras de transformação do caso de uso para o Event-B. Fonte o autor.

6. Trabalhos Relacionados

Existem muitos trabalhos que também propõem a formalização dos diagramas da para ajudar a tarefa de verificação durante o processo de desenvolvimento. Alguns trabalhos importantes como [6] e [7], apresentam métodos formais baseados na UML(rcos e KeY, respectivamente). No entanto não se baseiam em nenhuma metodologia conhecida que incentive seu uso. Outros trabalhos como [8] e [9] que fornecem um mapeamento da UML para linguagens formais reconhecidas (Object-Z e VDM++, respectivamente). Porém não há nenhuma menção a integrá-los em uma metodologia popular, como planejamos fazer.

7. Discussões e trabalhos em curso.

Neste trabalho, propusemos uma abordagem para inclusão de um método formal (Event-B) numa metodologia baseada em UML leve(ICONIX), a fim de trazer os benefícios da verificação formal para os profissionais da indústria. Nós já possuímos as regras de transformação dos diagramas da primeira fase do ICONIX implementadas.

Este trabalho tem alguns trabalhos em curso importantes. Um deles está sendo finalizado o mapeamento dos diagramas de robustez e do diagrama de sequencia, que pertencem a segunda fase do ICONIX. Após isto vamos realizar um experimento controlado com parceiros da indústria, que utilizam o processo ICONIX para verificar a viabilidade/usabilidade da nossa proposta para auxiliar a tarefa de verificação.

Referências

1. Abrial, J.R.: Modeling in Event-B: System and Software Engineering. 1st edn. Cambridge University Press, New York, NY, USA (2010)
2. Rosenberg, D., Stephens, M.: Use Case Driven Object Modeling with UML: Theory and Practice. Apress (2007)
3. Abrial, J.R., Butler, M., Hallerstede, S., Voisin, L.: An open extensible tool environment for Event-B. In: Proceedings of the International Conference on Formal Engineering Methods (ICFEM 2006). LNCS, Springer (2006) 588-605
4. Jacobson, I.: Object Oriented Software Engineering: A Use Case Driven Approach. Addison-Wesley (1992)
5. Snook, C., Butler, M.: UML-B: Formal modeling and design aided by UML. ACM Trans. Softw. Eng. Methodol. 15 (January 2006) 92-122
6. Chen, Z., Liu, Z., Stolz, V., Yang, L., Ravn, A.P.: A Refinement Driven Component-Based Design. In: Procs. of the 12th IEEE Intl. Conf. on Engineering Complex Computer Systems, Washington, USA, IEEE Computer Society (2007) 277-289
7. Ahrendt, W., Beckert, B., Hahnle, R., Schmitt, P.H.: KeY: A formal method for object-oriented systems. In: Procs. of 9th. Intl. Conf. on Formal Methods for Open Object-Based Distributed Systems, Cyprus, 2007. LNCS, Springer (2007)
8. Miao, H., Liu, L., Li, L.: Formalizing UML Models with Object-Z. In: Formal Methods and Software Engineering. Volume 2495 of LNCS. Springer (2002)
9. Lausdahl, K., Lintrup, H., Larsen, P.: Connecting UML and VDM++ with open tool support. In: FM 2009: Formal Methods. Volume 5850 of LNCS. Springer (2009)

Avaliação do Comprometimento das Equipes para o Impacto das Metodologias Ágeis na Utilização do Desenvolvimento de Software de uma Organização

José Carlos C. L. S. Filho[1], Willame P. Oliveira[2]

[1] Bacharel em Sistema de Informação – Faculdade das Atividades Empresariais de Teresina (FAETE)
Av. Dr. Nicanor Barreto, 4381 - Vale Quem Tem – 64057-355
Teresina – PI – Brasil

[2]Curso de Sistema de Informação – Faculdade das Atividades Empresariais de Teresina (FAETE)
Av. Dr. Nicanor Barreto, 4381 - Vale Quem Tem – 64057-355
Teresina – PI – Brasil

jcarloslimafilho@hotmail.com, wpowillame@gmail.com

Abstract. *This paper presents the results of a survey conducted in companies working with Software Engineering in State of Piauí-PI-Brazil, and applying a type of agile methodology in a project to develop a software product. The results obtained suggest that companies encourage the deployment of methodologies, but the teams have some resistance due to interference in current culture, and need more skills and knowledge of methodologies, for the successful management of product development is the crux to success of technology-based company.*

Resumo. *Este trabalho apresenta o resultado de uma pesquisa realizada com colaboradores de empresas que trabalham com Engenharia de Software no Estado do Piauí-PI-Brasil, e que aplicam um tipo de metodologia ágil em um projeto de desenvolvimento de software. Os resultados alcançados indicam que as empresas incentivam a implantação das metodologias, porém os times têm certa resistência devido à interferência na cultura atual, e precisam de mais qualificação e conhecimentos das metodologias, pois a bem-sucedida gestão de desenvolvimento de produtos é o ponto crucial para o sucesso de uma empresa de base tecnológica.*

1. Introdução

Existe hoje a necessidade de desenvolver *software* de forma mais rápida, mas com qualidade. Esse desenvolvimento pode ser obtido utilizando-se métodos ágeis e padrões organizacionais de processo. A popularização dos métodos ágeis ocorreu com o "Manifesto Ágil" (BECK et al., 2001).

Segundo o Standish Group (2011), projetos ágeis são três vezes mais bem sucedidos do que projetos não-ágeis de acordo com o relatório CHAOS 2012. Eles não informam quantos projetos estão em seu banco de dados, mas dizem que os resultados

são de projetos realizados entre 2002 e 2010.

Porém, para que a implantação de uma metodologia ágil tenha sucesso é necessário que as pessoas do time estejam motivadas e comprometidas com sua adoção (AGILE MANIFESTO, 2001). Ter membros do time realizando suas tarefas sem participar ativamente, não contribuindo com o processo, pode não trazer o resultado desejável para um bom desenvolvimento e/ou manutenção de um *software* (SOUZA NETO, 2010).

Existem diversos trabalhos relatando a implantação e adoção de métodos ágeis na literatura. (Version One, State of Agile Devolopment) (Claudia e Gisele, 2010) (Scrum e Xp direto das trincheiras). O nosso trabalha se diferencia pois se aprofunda na perspectiva dos participantes das equipes ágeis pesquisadas, buscando destacar como um membro do time se sente dentro da equipe, qual seu nível de conhecimento, o comprometimento com a metodologia e sua aplicação.

Dessa forma, o objetivo deste trabalho foi analisar o comprometimento das equipes que utilizam Metodologia Ágil (MA) no Estado do Piauí-Brasil. Essa análise foi feita com aplicação de um questionário buscando avaliar o empenho dos times ágeis em entender e exercer as atividades de metodologias ágeis adotadas por suas empresas, verificar se as equipes têm ambiente de trabalho colaborativo na execução das atividades, examinar o relacionamento e a confiança entre os integrantes das equipes, avaliar o nível de conhecimento das equipes com relação às metodologias utilizadas por suas empresas e analisar o resultado da implantação das metodologias perante a pontualidade das entregas e o *feedback* do cliente.

2. Implementação do questionário

Foi realizado um questionário disponibilizado via *on-line* utilizando a internet como meio de divulgação e execução. Os convites enviados para os participantes foram feitos através de *e-mail* e também através de *links* divulgados no grupo Ágile Piauí (grupo que incentiva práticas ágeis nas empresas que trabalham com Engenharia de *Software* no Estado do Piauí-Brasil).

O questionário, contendo o total de 26 questões, foi aplicado em um fornecedor mundial de soluções de questionário pela web (SurveyMonkey), e foi dividido em 4 (quatro) *seções* seguindo a seguinte ordem:

- Incentivo (Apoio) da empresa com 5 perguntas;
- Relacionamento com a equipe com 5 perguntas;
- Conhecimento da metodologia com 10 perguntas e
- Aplicação da metodologia hoje *(Feedback* do cliente) com 6 perguntas.

3. Análise e Resultados

A pesquisa envolveu 44 participantes, dos quais 31 foram selecionados por responderem o questionário por completo. Os participantes representavam 11 empresas diferentes e possuíam em média três anos de experiência com MA.

Mesmo com a pesquisa sendo divulgada em um grupo de metodologias ágeis aberto para todo o Estado do Piauí, os participantes que responderam o questionário

trabalham em empresas com sede na capital (Teresina). A pesquisa também não se preocupou em validar uma presença mínima de participantes por empresa, o que gerou um desbalanceamento, tendo empresa com apenas um participante e outras com mais de seis participantes, sendo este último, contendo a maioria dos entrevistados.

De acordo com a pesquisa, as metodologias usadas foram Scrum, Extreme Programming (XP), Kanban, Crystal Methods, Feature-Driven Development (FDD) e Scrumway, uma adaptação do Scrum feita por uma empresa participante. Entre essas, as mais usadas foram Scrum com 70%, Feature-Driven Development (FDD) com 26,67%, Extreme Programming (XP) e Kanban, ambas com 16,67% das respostas. Entre as pessoas entrevistadas se encontravam Desenvolvedor, Gerente de projetos, Engenheiro de requisito, Analista de negócio, Proprietário de classe, Scrum Master, Product Owner, Técnico de tecnologia da Informação, Analista de requisitos, Diretor de Tecnologia da Informação (TI) e Tester.

Nas seções a seguir são apresentadas algumas perguntas e seus resultados divididos pelos quatro tópicos do questionário. Toda a pesquisa realizada pode ser acessada por meio do link: http://jcarlos.site90.net/pesquisa.pdf.

3.1. Incentivo (apoio) da empresa:

Essa primeira sessão buscou identificar se o entrevistado se sente respaldado pela empresa com relação a ferramentas e ambiente de trabalho (maquinários adequados, materiais de escritório, ambiente climatizado com poucos ruídos no ambiente de trabalho).

A Figura 1 mostra na visão do entrevistado se o seu ambiente de trabalho pode, de alguma forma, contribuir para o melhor desempenho nas suas atividades diárias realizadas na empresa.

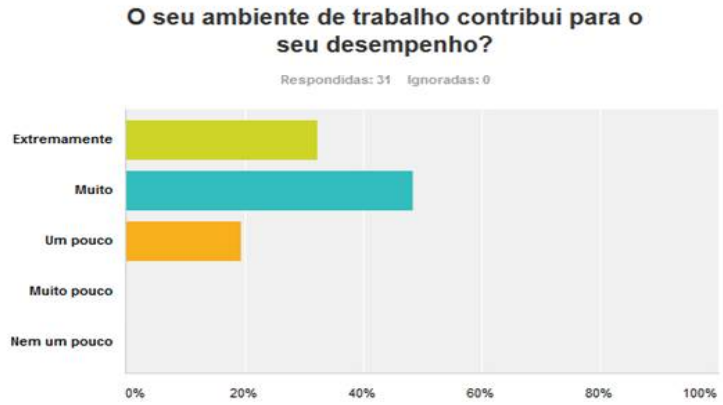


Figura 1. Ambiente de trabalho *versus* desempenho no trabalho.

Foi ressaltado o bom ambiente de trabalho nas empresas, 80,6% dos entrevistados afirmam ter um ambiente que contribui muito ou extremamente com seu trabalho. 19,4% contribuem um pouco e nenhuma menção de que o ambiente contribua muito pouco ou nem um pouco.

3.2. Relacionamento com a equipe

Com essa sessão, o trabalho procurou identificar se o entrevistado possui em seu

trabalho um espaço comunicativo, colaborativo e de confiança.

A Figura 2 mostra o relacionamento dos entrevistados com os seus gerentes ou superiores.

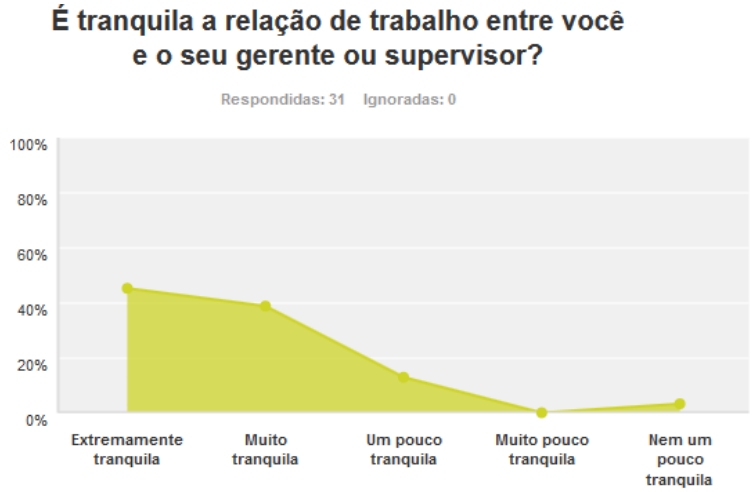


Figura 2. Relação de trabalho

Um bom relacionamento entre as hierarquias das empresas facilita a comunicação e transmissão de informações. Com 83,9% das respostas sendo Extremamente tranquila e Muito tranquila, se valida o bom relacionamento corporativo. A pequena percentagem 3,2% de Nem um pouco tranquila também valida o bom relacionamento.

3.3. Conhecimento da metodologia

Com essa sessão, o objetivo foi verificar como o entrevistado se compromete com a aplicação da metodologia ágil adotada na empresa, fazendo uma relação entre os treinamentos, tempo de experiência, nível de conhecimento, adaptação e utilização das atividades.

Essa seção foi baseada na relação de conhecimento com o comprometimento. A pesquisa mostra o pouco investimento em treinamento, segundo os entrevistados com 51,6% divididos em Pouco, Muito Pouco e Nenhum treinamento nas metodologias usadas (**Figura 3**).

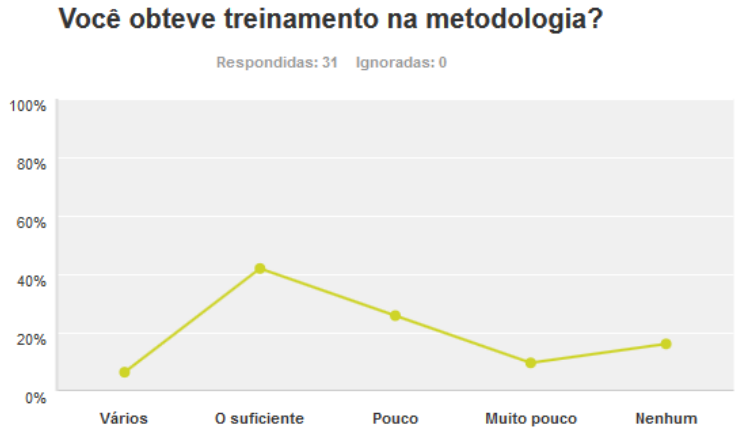


Figura 3. Treinamento sobre Metodologia Ágil

E ainda segundo os entrevistados, 32,26% indicam que as empresas usam menos

de 50% das atividades disponíveis pelos *frameworks*, 25,81% utilizam entre 50% a 70% das atividades disponíveis e pouco mais de 16% utilizam mais de 90% das atividades.

Alguns dos motivos, identificados na pesquisa, de somente a minoria utilizar mais de 90% das atividades, foram: Pouco tempo de adesão às metodologias ágeis, com equipes "jovens" se ajustando às atividades; e dificuldade de fazer estimativas das tarefas devido à experiência e conhecimento do negócio exigido.

3.4. Aplicação da metodologia hoje *(Feedback* do cliente)

Nessa sessão, o trabalho buscou identificar o comprometimento da equipe com os prazos indicados, e o resultado alcançado com o envolvimento e o *feedback* do cliente.

A filosofia defende a satisfação do cliente e a entrega de incremental prévio; equipes de projetos pequenas e altamente motivadas; métodos informais; artefatos de engenharia de *software* mínimos e, acima de tudo, simplicidade no desenvolvimento geral. Os princípios de desenvolvimento priorizam a entrega mais que a análise e projeto (embora essas atividades não sejam desencorajadas); também priorizam a comunicação ativa e contínua entre desenvolvedores e clientes (PRESSMAN, 2011). A Figura 4 mostra a participação dos clientes durante o desenvolvimento do projeto.

Figura 4. Participação ativa do cliente

A pouca participação do cliente foi constatada em mais de 50% das afirmações que indicaram entre Um Pouco e Nem um pouco (**Figura 4**). No entanto um ponto positivo citado pelos entrevistados é o alto nível de satisfação dos clientes indicando com 74,2%, que a maiorias dos *feedbacks* são positivos. Somente 3,2 % foi considerado negativo e 19,4% intercala entre positivo e negativo.

Verifica-se nos pontos positivos, a evidencia de algumas vantagens da MA que é o aumento do controle por parte dos gestores, uma vez que se baseia no que está realmente a ser produzido e no que vai ser feito em curto prazo. Como tal, há menos especulação, há mais visibilidade e adequação das medições e avaliações do estado das funcionalidades e tarefas realizadas (TOMÁS, 2009).

4. Considerações Finais

Esta pesquisa mostrou haver um bom ambiente de trabalho e um bom relacionamento entre as equipes, o que é fundamental para a motivação e comprometimento do indivíduo. Porém, falta mais treinamento (educação) de

colaboradores. 51,6% dos entrevistados indicam Pouco, Muito Pouco ou Nenhum treinamento sobre as metodologias utilizadas. Uma dificuldade constatada foi a pouca participação dos clientes (menos de 50%). No entanto, o que está sendo entregue ao longo das interações tem recebido (mais de 70%) um *feedback* positivo.

Como trabalhos futuros, pretende-se ampliar a pesquisa para outros Estados, aumentando a diversidade de empresas, cultura, pessoas e experiências analisadas. E ainda realizar revisões sistemáticas voltadas para os principais problemas levantados na implantação de metodologias ágeis pelas empresas, com o intuito de fornecer soluções para os problemas.

Referências

Agile Manifesto (2001). Disponível em http://agilemanifesto.org/, acessado em 02 de Setembro de 2013.

Alves, S.R e Luiz, A. A. (2013) "Engenharia de requisitos em metodologias ágeis". Disponível em: http://www.cpgls.ucg.br/Arquivos, acessado em maio de 2013.

Beck, K. (2001). "Manifest for agile software development". 2001. Disponível em http://www.agilemanifesto.org.acessado em maio de 2013.

Melo, C. O e Ferreira, G. R. M. (2010). Adoção de métodos ágeis em uma Instituição Pública de grande porte - um estudo de caso. Disponível em: http://www.agilebrazil.com/2010/pt/wbma2010.pdf, acessado em outubro de 2013.

Pressman, R. S. (2011) "Engenharia de Software: Uma abordagem profissional". Bookman, Porto Alegre.

Souza Neto, O. N. (2010). Análise Comparativa das Metodologias de Desenvolvimento de Softwares Tradicionais e Ágeis. [Dissertação–Mestrado]. Belém: Universidade da Amazônia, 2010.

The CHAOS Manifesto, The Standish Group. (2012). Agile Succeeds Three Times More Often Than Waterfall. Disponível em http://www.mountaingoatsoftware.com /blog/agile-succeeds-three-times-more-often-than-waterfall, acessado em outubro de 2013.

Tomás, M. R. S. (2009). "Métodos ágeis: características, pontos fortes e fracos e possibilidades de aplicação". IET Working Papers Series No. WPS09.

Análise e Implementação de um Sistema para o Gerenciamento de Estágios Curriculares na Faculdade Piauiense - FAP/Parnaíba*

Ely N. Barros[1], Mayllon V. da Silva[1]

[1]Bacharelado em Sistemas de Informação – Faculdade Piauiense – FAP/Parnaíba
CEP 64.202-260 – BR 343 – Parnaíba – PI – Brasil

{elybarros,mayllonveras}@gmail.com

Abstract. *The supervised internship is one of the mechanisms where it is possible to combine theory and practice, giving to the academics learning and experience that prepares them to face the world beyond university. The teaching institutions need to have the control of their internships curriculums and many documents are produced during the student's training. Concerned about this, our work aims to help the coordination of the Computing course from Faculdade Piauiense FAP/Parnaíba, to manage their internships curriculums. Many factors that need optimization through computerized processes were identified and to achieve this, it was applied a software development process called Rational Unified Process, which provided defined processes to achieve the objectives proposed.*

Resumo. *O estágio supervisionado é um dos mecanismos em que se é possível aliar teoria e prática, dando ao acadêmico a aprendizagem e experiência que o preparará para enfrentar o mundo fora da universidade. As Instituições de Ensino precisam ter o controle de seus estágios curriculares, pois diversos documentos são produzidos durante o estágio dos alunos. Diante disso o presente trabalho visa auxiliar a coordenação do curso de Sistemas de Informação da Faculdade Piauiense FAP/Parnaíba, e a gerenciar seus estágios curriculares. Diversos fatores que necessitam de otimizações através de processos informatizados foram identificados. Para que esses fatores pudessem ser otimizados, foi aplicado um processo de desenvolvimento de software chamado Rational Unified Process, que forneceu processos definidos para alcançar os objetivos propostos.*

1. Introdução

No universo acadêmico o estágio representa uma chance a mais de o estudante entrar para o mercado de trabalho. A teoria aprendida dá ao aluno uma base para que este possa transformá-la em prática. O estágio supervisionado é um dos mecanismos que se é possível aliar teoria e prática dando ao acadêmico a aprendizagem e experiência que o preparará para enfrentar o mundo fora da universidade.

Baseando-se nos argumentos supracitados, torna-se relevante que a instituição

* Artigo produzido a partir do trabalho de conclusão de curso na Faculdade Piauiense, 2012.

de ensino tenha um controle mais definido de seus estágios, sendo esta uma das principais ferramentas para o sucesso de seus discentes no mercado de trabalho. Acerca disso, identifica-se que a Faculdade Piauiense – FAP/Parnaíba atualmente não possui um controle informatizado sobre seus estágios.

Partindo dessa premissa, a presente pesquisa tem como proposta uma análise da situação atual dos estágios e o desenvolvimento de um sistema informatizado como solução, para gerenciar e controlar os estágios desta IES, abrangendo inicialmente o curso de Sistemas de Informação, sendo este o curso que demostrou maior interesse. O objetivo principal é poder fornecer aos gestores uma solução para o acompanhamento sistematizado dos estágios curriculares.

Para alcançar os objetivos, foi realizado uma pesquisa exploratória com levantamento de dados bibliográficos e entrevistas com o Coordenador do Curso, que tem uma vasta experiência nos processos atuais. Foi realizado também uma estudo de caso, observando-se o ambiente sem provocar alterações, dessa forma, coletando informações confiáveis sobre o processo do estágio. Foi utilizado o RUP como processo de desenvolvimento de software. Para implementação foi utilizado a plataforma Java para Web, utilizando-se também dos *frameworks hibernate* e *primefaces* como auxilio. Padrões de projetos e uma arquitetura de software em três camadas foram utilizados para facilitar futuros incrementos.

Este artigo está dividido em cinco seções. A primeira, introdução, apresenta uma contextualização, expõe os objetivos, o problema identificado e a proposta de solução. A segunda seção expõe sobre o estágio na educação superior, sua obrigatoriedade e sua importância. Na próxima seção é focado mais no estágio na Faculdade Piauiense, seu regimento interno, seus regulamento, e finalmente o regulamento do estágio do curso de Sistemas de Informação. Na quarta seção é encontrado os métodos e ferramentas utilizadas para o desenvolvimento do software. Por fim, na última seção é apresentado as considerações finais, demonstrando como o software projetado pode ajudar a gestão dos estágios.

2. O Estágio na Educação Superior

Uma instituição de ensino superior tem um papel fundamental na sociedade: o de integração dos jovens no mercado de trabalho. Desse modo, as universidades têm a responsabilidade de torná-los cidadãos capazes de exercerem suas respectivas funções em diversas áreas de conhecimento. Para tal, elas oferecem aos estudantes subsídios não apenas teóricos, mas também práticos, de maneira que prepare estes, de forma adequada, para o exercício de sua função.

Com este intuito as instituições de ensino utilizam-se de ferramentas para unir o conhecimento teórico ao prático. Assim, as faculdades incluem em seu projeto pedagógico o estágio como obrigatório para conclusão do curso superior.

Assim, é no ambiente de trabalho proporcionado pelo estágio que o estudante terá oportunidade de se desenvolver profissionalmente, a partir de atividades do próprio dia a dia, enfrentando novos desafios, se envolvendo com profissionais da área e adquirindo novos conhecimentos.

O estágio é regido pela Lei 11.788/2008 (BRASIL, 2008), e em seu artigo 3º é afirmado que não existe vínculo empregatício no estágio. Para isso, no entanto, deve ser observados os seguintes requisitos: matrícula e frequência regular do educando; celebração de termo de compromisso entre o educando, a parte concedente do estágio e a instituição de ensino e, por fim, a compatibilidade entre as atividades desenvolvidas no estágio e aquelas previstas no termo de compromisso. Sendo que qualquer descumprimento de um desses requisitos caracteriza-se vínculo empregatício.

3. As Etapas do Estágio na Faculdade Piauiense

A FAP conta com alguns documentos internos que regem sobre o estágio de seus discentes. O Capítulo IX do Regimento Interno da FAP (FAP, 2007) dispõe sobre os estágios. Segundo o art. 121, desse mesmo capítulo, os estágios obedecem regulamentos próprios, um para cada curso, de acordo com o Conselho de Ensino, Pesquisa e Extensão – CEPEX .

O regulamento do estágio supervisionado do curso de Sistemas de Informação (FAP, 2011) cita, em seu art. 2º, parágrafo 2º, que o estágio neste mesmo curso é composto por duas disciplinas de 160 horas, cada uma, devendo ser cumpridas nos blocos VII e VIII.

O estágio do Curso de S.I conta basicamente com 6 personagens: o Coordenador de Curso; o Coordenador Geral de Estágio; o Professor Supervisor da Disciplina; o Professor Orientador de Estágio; o Professor Supervisor de Campo/Empresa e o Aluno Estagiário.

O estágio é realizado junto a empresas privadas ou públicas, sendo ele, gerido pelos professores das disciplinas de estágio supervisionado I e II. O aluno terá seu estágio supervisionado por um professor orientador, um professor da disciplina e mais o supervisor de campo. Os professores são indicados pelo coordenador de curso, já o supervisor de campo é indicado pela empresa.

O Coordenador Geral de Estágio tem como atribuição gerir o estágio, dessa forma ele deve realizar contratos de convênios; manter e divulgar convênios; orientar alunos sobre a lei do estágio e emitir apólices de seguro. O professor da disciplina é responsável por examinar toda a documentação apresentada e efetuar os lançamentos no diário de classe quanto à frequência e as notas. Já o professor orientador é aquele responsável por orientar e examinar o plano de estágio produzido pelo aluno. E por fim, o aluno tem como suas principais obrigações a elaboração do plano de estágio como também a entrega dos relatórios necessários dentro dos prazos estipulados.

4. Sistema de Gerenciamento de Estágios (SGE)

O sistema tem como proposta ajudar os gestores no processo de gerenciamento dos estágios, fornecendo meios de controle dos documentos entregues e de suas situações corrente, meios de saber a situação dos estágios de cada aluno e gerando relatórios que possam ajudar em tomada de decisões. Através do SGE (BARROS, 2012) também é possível que os alunos transfiram seus relatórios de atividades e planos de estágios eletronicamente, evitando impressões desnecessárias e desencontros com seus respectivos orientadores. Algumas IES já possuem um modulo próprio para controle de

estágios. É caso por exemplo, da Pontifícia Universidade Católica de Minas Gerais e da Universidade Federal de São Paulo. Isso reforça a importância de um sistema como esse para o controle dos estágios.

Como processo de desenvolvimento de software foi utilizado o modelo Rational Unified Process (RUP). O RUP é um processo iterativo e interativo, ele é configurável para cada equipe e divide-se em quatro fases: concepção, elaboração, construção e transição. Segundo Kroll, Kruchten (2003, p.3), o RUP é um processo bem definido e estruturado que define claramente quem é o responsável pelo que, como as coisas são feitas, e o quando fazê-las.

Na fase de concepção, foi a fase onde houve uma maior obtenção de requisitos. Nessa fase foi feito levantamento dos requisitos junto ao cliente e a documentos, dessa forma tentando levantar o maior número de requisitos possíveis e ter uma visão mais ampla do sistema. Após o levantamento desses requisitos foi feito uma organização dos mesmos, sendo estes classificados em casos de uso, consultas ou cadastros. Ao final da fase de concepção foi feito um planejamento, associando os diferentes casos de usos a ciclos iterativos de desenvolvimento.

Alguns elementos foram priorizados a fim de que os processos de maiores riscos fossem tratados primeiro. Após os elementos com maiores riscos serem tratados, o sistema deve ter um ritmo de desenvolvimento mais rápido pelo fato do desenvolvedor ter maior conhecimento do sistema e, claro pelos outros processos serem mais simples de resolver. Desse modo, o desenvolvedor também encontra possíveis problemas logo no início do desenvolvimento.

Tabela 1. Planejamento dos ciclos iterativos

Ciclo	Casos de Uso	Manutenção de Informações	Consultas	Obs.	Esforço estimado
1	Plano de estágio (40)	Estágio (10), Aluno (5), Coord. de estágio (5), Prof. Orientador (5), Prof. Da disciplina (5)	-	-	70 horas
2	Termo de compromisso (18), Convênio (18)	Convênio (8), Atividade de estágio (6)	-	-	50 horas
3	Gerar relatórios semanais (19)	Frequência (11)	-	-	30 horas
4	Gerar relatórios mensais (19)	-	-	-	19 horas
5	Entrega da frequência (21)	-	-	-	21 horas
6	Ficha de acompanhamento (22)	Supervisor de campo(5)	Todas(11)	-	38 horas

Na Tabela 1 é possível verificar os ciclos criados priorizando os casos de usos

com maiores riscos. Os casos de maiores riscos foram alocados nos ciclos 1 e 2, eles também obtiveram a maior pontuação de esforço estimado (medido em horas). As consultas foram todas alocadas no último ciclo, pois já que são apenas extrações de informações já inseridas elas representam um menor risco para o desenvolvimento.

A partir do planejamento foi realizado um detalhamento maior, organizando-se as atividades a serem realizadas em cada ciclo e quanto de esforço está previsto para cada atividade. Feito isso foi iniciado a execução dos ciclos iterativos que ocorrem entre as fases de elaboração e construção. Durante os ciclos iterativos foram expandidos os casos de usos antes não detalhados. Foram também produzidos diagramas de sequências e diagramas conceituais mais completos.

Para o desenvolvimento foi utilizado o Java como linguagem de programação. Os princípios da programação orientada a objeto junto a padrões de projetos e a aplicação de alguns *frameworks* tornou possível manter a organização e manutenibilidade do sistema. O *PrimeFaces* foi escolhido como *framework* front-end devido a sua vasta opções de componentes web e sua documentação de qualidade. O *PrimeFaces* tem um grande diferencial em relação a produtos de fornecedores pois ele é utilizado pelos desenvolvedores nos projetos de seus clientes *(PRIMEFACES).* Para a persistência dos dados foi utilizado o *framework hibernate*, o que permitiu um aumento da produtividade já que o mapeamento objeto-relacional foi feito pelo próprio *framework.*

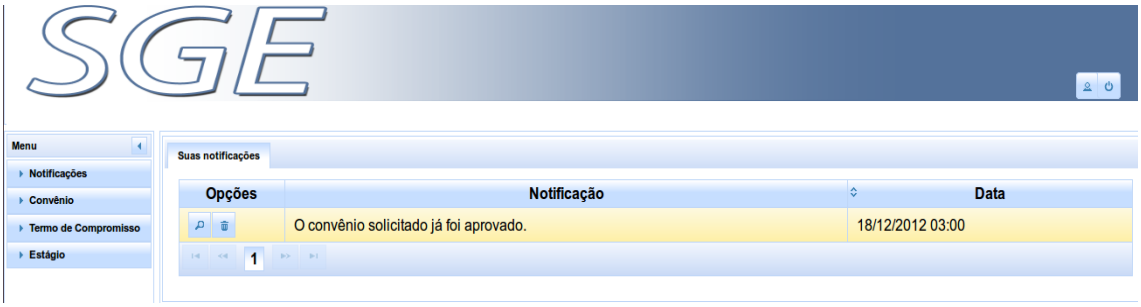


Figura 1. Tela inicial do SGE

Na tela inicial do sistema, Figura 1, é possível observar algumas funcionalidades do sistema. Logo quando acessado o sistema já trás as notificações do usuário logado. Ele pode visualizar essa notificação clicando no ícone da lupa ou excluí-la clicando no ícone do lixeiro. O menu notificações é o que está sendo exibido na imagem, o meu convênios dá ao usuário a possibilidade de solicitar, efetivar ou consultar convênios, dependendo de seu papel no sistema. O menu termo de compromisso trás as mesmas opções que o de convênio, mas agora trata-se de um termo de compromisso. O menu estágio é possível controlar documentos, iniciar um estágio, gerar relatórios, gerenciar o plano de estágio, enfim, lá estão reunidas todas as funcionalidades relacionadas ao estágio em si. Na funcionalidade de avaliar convênios, Figura 2, o Coordenador de Estágio recebe uma solicitação, clica em “Visualizar notificação” e depois de sua avaliação decide se aprova ou solicita uma revisão. Na “Lista de convênios” ficam todos os convênios solicitados, se o sistema for acessado pelo avaliador. Se for acessado pelo solicitante, ficam apenas seus convênios e suas respectivas situações no processo.

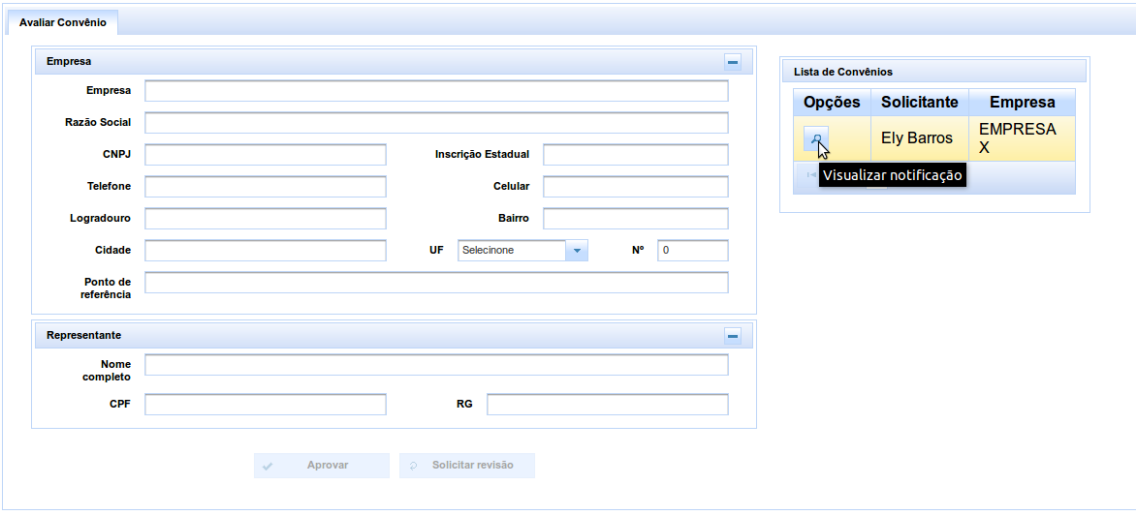


Figura 2. Tela de cadastro/avaliação de convênio

5. Considerações Finais

O projeto do software fornece base suficiente para que os estágios sejam acompanhados sistematicamente. O software foi projetado a fim de facilitar a geração dos relatórios semanais e mensais referentes ao estágio. Pelo sistema, tendo as informações necessárias, com apenas um clique é possível gerar os relatórios semanais. Para os mensais basta que seja informado um texto corrido com o apanhado geral das atividades feitas durante o mês e com um clique o relatório também pode ser gerado.

Para que fosse projetado um software de fácil implementação e manutenção, foi necessário o estudo de um processo de desenvolvimento de software e a aplicação de padrões de projetos. Estes padrões foram essenciais, através deles melhorias claras foram identificadas durante o desenvolvimento do projeto.

Para trabalhos futuros deve ser feito a implantação do sistema para testes no ambiente de trabalho. Pode também ser feito uma avaliação entre dois semestres com uma turma, sendo que no segundo semestre deve-se utilizar o sistema para que seja feito uma pesquisa dos possíveis benefícios gerados na utilização do SGE. Deve-se acrescentar também possíveis novas funcionalidades que venham a otimizar o processo, o gerenciamento dos planos de estágios sem a necessidades de *uploads*, por exemplo, poderia gerar um controle ainda mais efetivo.

Referências

BARROS, E. N. Sistema de Gerenciamento de Estágios, 2012.

BRASIL. Lei 11.788/2008.

FACULDADE PIAUIENSE - FAP. Regimento Interno da FAP, 2007.

FACULDADE PIAUIENSE - FAP. Regulamento do Estágio Supervisionado do Curso de Sistemas de Informação, 2011.

KROLL, Per; KRUCHTEN, Philippe. Rational Unified Process Made Easy: a practitioner's guide to the RUP. Addison Wesley , 2003.

Desenvolvimento de um Sistema de Controle de Iluminação Fuzzy

Jerônimo Vianney Pereira Sousa[1], Aldir Silva Sousa[1]
[1] Faculdade de Ciências e Tecnologia do Maranhão
Rua Aarão Reis, 1000 - Centro · Caxias, MA – Brazil
jvps.jvps1@gmail.com, aldirss@yahoo.com.br

Abstract. *This paper describes the process of developing a lighting control system that uses concepts of intelligent systems in their modeling. The purpose of this paper is to demonstrate the advantages of designing lighting control systems through using computational intelligence techniques. The lighting control system proposed in this paper has been developed by applying a fuzzy controller to sensors coupled to the Arduino platform. In tests, it was possible to verify the efficiency of the proposed method since it was able to increase the lifetime of used batteries in 159 percent.*

Resumo. *Este trabalho descreve o processo de desenvolvimento de um sistema de controle de iluminação que utiliza conceitos de sistemas inteligentes em sua modelagem. O objetivo deste artigo é demonstrar as vantagens de se projetar um sistema de controle de luminosidade através de abordagens próprias da inteligência computacional. O sistema de controle de luminosidade proposto neste trabalho foi desenvolvido aplicando um controlador fuzzy a sensores acoplados à plataforma Arduino. Em testes realizados, pôde-se verificar a eficiência do método aqui proposto já que foi possível aumentar o tempo de vida de pilhas utilizadas em 159 por cento.*

1. Introdução

A automação tem se mostrado um meio eficaz na busca por eficiência energética no âmbito residencial. A automação provê vários mecanismos de controle separados por funções específicas, naturalmente pertencentes ao contexto residencial, tais como controle de acesso e segurança eletrônica, controle de iluminação, controle de climatização, controle de acionamento de eletrodomésticos, entre vários outros.

Sistemas de controle de iluminação constituem a melhor solução para o problema do uso negligente da iluminação artificial no contexto residencial. Canato (2007) descreve algumas formas de implementação comumente encontradas em sistemas de iluminação automatizados.

Os sistemas cientes de contexto estão entre algumas soluções que podem atender a demanda por recursos de forma completamente autônoma, antecipando-se às necessidades do usuário sem a interferência direta do mesmo [Satyanarayanan 2001]. O sistema aqui proposto visa diminuir desperdício de energia controlando a potência utilizada por luminárias e balanceando o grau de luminosidade das lâmpadas de acordo com o aproveitamento da luz natural no interior dos ambientes.

2. Controlador *Fuzzy* Para Iluminação

No desenvolvimento do projeto do controle de iluminação baseou-se na lógica nebulosa para concepção de um cerne inteligente para o sistema. Esta unidade é responsável pela decisão sobre que saída deve ser gerada como sinal de controle para um circuito de potência.

Aplicações inteligentes, voltadas para o aprendizado do comportamento de uma casa ou de seus habitantes, podem constituir a base de um sistema de controle de iluminação, assim como sistemas que abordam o tratamento de contextos que envolvem incerteza, como é o caso da lógica nebulosa [Vainio, Valtonen e Vanhala 2006]. Em um problema de controle de iluminação um dos parâmetros a serem considerados para a modelagem de um controlador *fuzzy*, é a intensidade luminosa em um ou vários pontos do ambiente.

Neste trabalho adotamos a intensidade de iluminação no ambiente como único parâmetro, a partir do qual devam ser tomadas todas as decisões do controlador *fuzzy* projetado.

Processo de desenvolvimento do controlador fuzzy

O primeiro passo no processo de desenvolvimento do controlador foi definir quais variáveis, físicas ou não, atuantes sobre o ambiente, deve-se adotar na modelagem do sistema. Em [Mou-Lin e Ming 2009] é descrito um controlador nebuloso para controle da luminosidade ambiente, que usa tanto o método *DISO, Double Input Simple Output*, quanto o *SISO, Simple Input Simple Output*. Controladores *SISO* possuem apenas uma variável na entrada e uma na saída, enquanto que os do tipo *DISO*, possuem duas variáveis na entrada. No referido trabalho, as variáveis de entrada adotadas são a intensidade luminosa no ambiente e a taxa de variação da luz, com apenas três conjuntos nebulosos cada uma. A mesma estratégia pode ser observada em [Chenghui, Naxin, Maiying e Zhaolin 2005], onde os conjuntos nebulosos *Daytime* e *Night* são definidos para a variável *Lighting* e, assim como [Mou-Lin e Ming 2009], uma variável referente à taxa de variação da luminosidade, *DeltaL*, com conjuntos MB, MS, Zero, PS e PB, é adotada.

Neste trabalho optou-se por se projetar um controlador nebuloso do tipo *SISO, Simple Input Simple Output*, com uma variável de entrada e uma de saída.

Para a construção do hardware projetado optou-se por se utilizar uma plataforma de prototipagem eletrônica microcontrolada, no caso, a placa *Arduino*. *Arduino* é uma plataforma de prototipagem eletrônica de código aberto baseada em *hardware* e *software*, flexível e fácil de usar.

O *Arduino* possui um ambiente de desenvolvimento de aplicações que embute uma ferramenta para leitura e escrita de dados seriais, o *Serial Monitor* [McRoberts 2010]. Através do *Serial Monitor* fora possível visualizar os dados coletados pelo sensor. Os conjuntos nebulosos definidos para a variável Iluminação foram: ME, ES, NM, CL e MC, descritos, com seus respectivos universos de discurso, na Tabela 1 e Fig. 1, abaixo.

TABELA I. CONJUNTOS NEBULOSOS DO CONTEXTO LUMINOSIDADE

Luminosidade do Ambiente(x)			
Conjunto	***Rótulo***	***Universo de discurso***	
Muito Escuro	ME	x < 200	
Escuro	ES	0	400
Normal	NM	200	600
Claro	CL	400	800
Muito Claro	MC	x > 800	

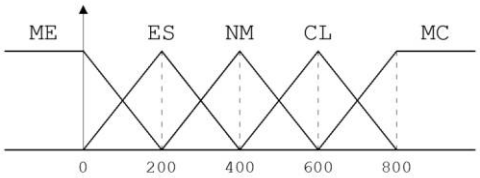


Fig. 1. Conjuntos nebulosos da variável *Iluminação.*

Para definição dos limites superior e inferior do universo de discurso da variável referente à potência elétrica, levaram-se em consideração os valores, mínimo e máximo, aceitáveis para regulação da largura de pulso, no *Arduino*. Para a modulação da largura de pulso, *PWM*, ou seja, do tempo em que o sinal deve se manter em 5v, durante um ciclo do período regular, a placa *Arduino* aceita valores entre 0 e 255 [Hirzel 2013].

A Fig. 2 mostra a faixa de valores aceita como parâmetro da função *analogWrite()* do *Arduino* e a porcentagem do clico a que corresponde o tempo em que o sinal se encontra em nível alto.

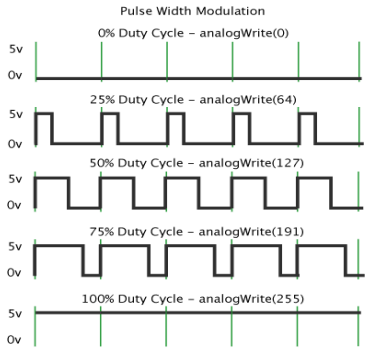


Fig. 2. Detalhe da modulação por largura de pulso, *PWM*, da placa *Arduino.*

Fonte: [12]

Os conjuntos nebulosos definidos para a variável *Potência* foram: MN, BX, NM, AL e MX, descritos, com seus respectivos universos de discurso na Tabela 2 e a Fig. 3, abaixo.

TABELA II. CONJUNTOS NEBULOSOS DO CONTEXTO POTÊNCIA

Potência			
Conjunto	***Rótulo***	***Universo de discurso***	
Mínima	MN	-63.75	63.75
Baixa	BX	0	127.5
Normal	NM	63.75	191.25
Alta	AL	127.5	255
Muito Alta	MX	191.25	318.75

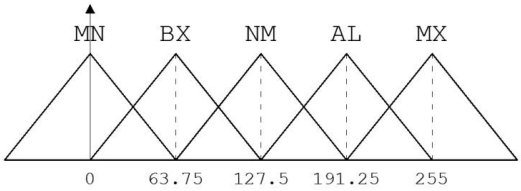


Fig. 3. Conjuntos nebulosos da variável *Potência.*

Como o controlador desenvolvido neste trabalho é do tipo *SISO*, as regras *fuzzy* definidas para seu conjunto de regras utilizam apenas uma variável de entrada, no antecedente, e uma variável de saída, no consequente.

A base de conhecimento do controlador pode ser resumida conforme as regras da Tabela 3.

TABELA III. CONJUNTO DE REGRAS DO CONTROLADOR *FUZZY*

x				
ME	***ES***	***NM***	***CL***	***MC***
MX	AL	NM	BX	MN

Circuito do protótipo

Para realização dos testes do controlador, fora construído um protótipo em escala reduzida, onde foram acopladas duas pequenas lâmpadas incandescentes, do tipo foquito, comumente utilizadas em lanternas. Cada uma dessas lâmpadas, *L1* e *L2*, conforme a Fig. 5, foram alimentadas por conjuntos de pilhas distintos, *B1* e *B2*, de 6v, cada um. Além desses componentes, percebe-se na Fig. 5, dois transistores, *Q1* e *Q2*, utilizados no controle das lâmpadas, alguns resistores para adequação da corrente e divisão de tensão, *R1*, *R2*, *R3*, *R5*, *R6* e *R7* e um sensor *LDR*, conectado a um dos pinos analógicos da placa, *A0*.

Pode-se notar, também, na Fig. 5, a utilização de alguns pinos digitais do *Arduino*, rotulados por *D8* e *D9*, para mensuração dos níveis de tensão dos conjuntos de pilhas, e *D10* e *D11*, para controle das tensões de base dos transistores *Q1* e *Q2*.

Cada uma das lâmpadas são controladas por controladores embutidos no *Arduino*, sendo um do tipo ON/OFF e o outro do tipo *fuzzy*.

O controlador ON/OFF fora projetado para acionar a primeira lâmpada, *L1*, assim que a placa recebesse o valor 300, através de uma função de leitura sobre o pino *A0* do *Arduino* (Sensor). Já, o controlador *fuzzy* fora projetado para controlar a tensão aplicada à base do transistor *Q2*, controlando a intensidade da lâmpada *L2*, através de um sinal regulado de acordo com o seu conjunto de regras. O sinal aplicado à base do transistor

Q2 varia de acordo com a saída *fuzzy*, criando um efeito de enfraquecimento gradual do brilho da lâmpada à medida que o sensor detecta um maior nível de luz no ambiente e aumentando o seu brilho quando ocorre o contrário. A Fig. 4 esboça o circuito projetado.

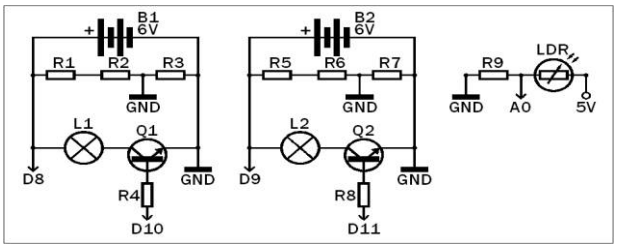


Fig. 4. Esquema do circuito do protótipo construído

3. Resultados

O controlador nebuloso de iluminação projetado respondeu mais eficientemente que o controlador ON/OFF, em termos de economia de energia. A Tabela 4 mostra o tempo que os conjuntos de pilha, *B1* e *B2*, passaram até se descarregarem, usando os dois sistemas.

Os experimentos foram finalizados quando todas as lâmpadas se encontravam apagadas por consequência do descarregamento de seus respectivos conjuntos de pilhas. Nesse caso, o último conjunto de pilhas a se descarregar fora o *B2*, que alimentava o foquito gerido pelo controlador *fuzzy*.

A Tabela 4 resume os resultados obtidos.

TABELA IV. RESULTADO DOS TESTES REALIZADOS

Teste	***Duração do conjunto pilhas sem o controlador ON/OFF***	***Duração do conjunto pilhas com o controlador fuzzy***
01	8100s	12900s
02	8160s	21120s

7. Conclusões

Este trabalho propõe a aplicação de controladores *fuzzy* para controle eficiente de energia, projetando-o de forma a aproveitarem os níveis da luminosidade natural, balanceando a potência de componentes como lâmpadas, abajures, arandelas, ou outro tipo similar, de acordo com a incidência da luz do dia no interior dos ambientes.

O sistema desenvolvido neste trabalho respondeu mais eficientemente que os controles do tipo ON/OFF, mostrando-se mais adequados que esses para aplicações de otimização do uso da eletricidade.

Uma vez que o sistema fora projetado para responder de forma completamente autônoma e independente das ações do usuário, conclui-se que o esse constitui uma solução promissora para agregação em sistemas de computação ubíqua ou pervasiva, automação residencial ou aplicações de domótica. Isso se deve ao fato de que através do

sensor utilizado para captação da luminosidade ambiente, o sistema ser capaz de perceber alterações de contexto referentes às variações na luz, o que coincide com as características de sistemas de computação ciente de contexto.

Referências

Canato, D. A. (2007) Utilização de Conceitos de Integração de Sistemas Direcionados à Domótica – Estudo de Caso para Automação Residencial, Universidade Estudal de Campinas, Campinas.

Satyanarayanan, M. (2001) Pervasive computing: Vision and challenges.Personal Communications, IEEE, v. 8, n. 4.

Vainio A. M., Valtonen, M. e Vanhala, J. (2006). Learning and adaptive fuzzy control system for smart home. In: Mana, A. et al. (eds.) Developing Ambient Intelligence, The First International Conference on Ambient Intelligence Developments. Setembro, 2006. Sophia Antipolis, France.

Mou-Lin, J. e Ming, C. H. (2009) Labview-based fuzzy controller design of a lighting control system. Journal of Marine Science and Technology, vol. 17.

Chenghui, Z.; Naxin, C; Maiying Z. e Zhaolin, C. (2005) Application of Fuzzy Decision in Lighting Control of Cities, 44h IEEE Conference on Decision and Control, and the European Control Conference 2005, Seville, Spain.

McRoberts (2010). Beginning Arduino. Apress.

Hirzel, T. (2013) PWM. Disponível em: < http://arduino.cc/en/Tutorial/PWM> Acessado em: 19 de maio de 2013.

CONTROLE E GERENCIAMENTO DO CONSUMO DE ENERGIA UTILIZANDO ARDUINO

Jaderson E. R. Costa[1],Raimundo P. C. Neto[2]

[1]Centro de Ensino Unificado de Teresina–(CEUT)
Teresina – Piauí

[2]Centro de Ensino Unificado de Teresina – CEUT
Teresina – Piauí
jadersonerocha@hotmail.com, netocunhathe@gmail.com

Abstract.The management of energy resources is one of the biggest problems faced by today's modern society. The article presents an idea of prototype that uses the concept of home automation / building, for triggering electronic devices and managing the consumption of energy spent per unit through the use of sensors and actuators. For this project we used the Arduino electronics prototyping platform, so that you can activate, manage and monitor their energy costs through the use of sensors like the ACS-712 (sensor adjustable current) sensors, actuators and communication, as lamps, televisions, air, among others. The user control is accomplished through wired or wireless communication and can make use of various devices that will enable Access to the internet. This work proposes the development of a Wireless Sensor Network (WSN), in order to monitor electronic equipment, allowing the user to control anyd evice.
Keywords: Wireless Sensor Networks, Arduino, Energy and Automation.

Resumo. A gestão de recursos energéticos é um dos grandes problemas que hoje é enfrentado pela sociedade moderna. O artigo apresenta uma ideia de protótipo que utiliza o conceito de automação residencial/predial, para o acionamento de dispositivos eletrônicos e o gerenciamento do consumo de energético gasto por cada aparelho através do uso de sensores e atuadores. Para tal projeto utilizou-se da plataforma de prototipagem eletrônica arduino, que que assim possa acionar, controlar e monitora seus gastos de energia através do uso de sensores como o ACS-712 (sensor de corrente ajustável), sensores de comunicação e os atuadores, como lâmpadas, televisores, aparelhos de ar, entre outros. O controle pelo usuário é realizado através de comunicação cabeada ou sem fio, podendo fazer uso de variados aparelhos que lhe possibilite o acesso à internet. Este trabalho propõe o desenvolvimento de uma Rede de Sensores sem Fio (RSSF), com o objetivo de monitorar equipamentos eletrônicos, permitindo ao usuário o controle sobre qualquer aparelho.
Palavras-chaves: Redes de Sensores sem Fio, Arduino, Energia e Automação.

1. Introdução

A escassez de recursos naturais em diversas áreas do planeta, fez surgir o conceito de Tecnologia da Informação (TI) verde, com o objetivo de buscar soluções para evitar a problemas maiores pela ausência desses recursos. Dentre as tecnologias que vem apoiando como alternativa de facilitar a implementação de projetos em TI verde, podemos destacar a computação pervasiva, através de redes sensores sem fios.

Este projeto demonstra a utilização dos conceito de automação residencial/predial para o controle e o monitoramento dosaparelhos eletrônicos que são

utilizados em uma residênciapodendo assimgerenciar o consumo de energia, dando ao usuário uma maior clareza sobre seu gastos em kw/h e o valor a ser pago o que fornece ao consumidor uma fonte mais detalhada sobre seu uso e consequentemente lhe fornece meios de contribuir com a diminuição de possíveis desperdício desnecessários, além de alternativas para determinação de padrão de consumo, bem como o gerenciamento dos equipamentos, através da utilização de RSSF.

A proposta teve como objetivo desenvolver um sistema de controle e gerenciamento de energia eficiente e financeiramente viável para o consumidor, fazendo uso de sensores em conjunto com o arduino.Na seção 2 é apresentado a fundamentação teórica deste trabalho, com os conceitos de Redes de Sensores sem fio. Na seção 3 será apresentado a placa Arduino. Na seção 4 será apresentado alguns trabalhos desenvolvidos na área. Na seção 5será apresentado arquitetura proposta. Na seção 6 descreveremos sobre o protótipo. Na seção 6 será relatado os testes realizados. Na seção 7 demostraremos os testes realizados. Na seção 8 os resultados obtidose na seção 9 as considerações finais.

2. Redes de Sensores Sem Fio

As Redes de Sensores sem Fio (RSSFs) são tecnologia atualmente em expansão e em amplo crescimento. Segundo Loureiro at al, as redes de sensores sem fio se diferem das redes de computadores tradicionais em vários aspectos. Normalmente essas redes possuem um grande número de nodos1 distribuídos, tem restrição de energia, e devem possuir mecanismos para autoconfiguração e adaptação devido a problemas como falhas de comunicação e perda de nodos. Uma RSSF tende a ser autônoma e requer um alto grau de cooperação para executar as tarefas definidas para a rede.

As Redes de sensores sem fio consistem de um grande número de dispositivos sem fiosdistribuídos em uma região de interesse. Sensores têm conectividade sem fio e são conectados a uma rede, tal como a Internet. Eles são tipicamente alimentados por baterias com comunicação e funções de computação limitadas. Cada nó pode ser equipado com uma variedade de modalidades de sensoriamento tais como acústico, sísmico, e infravermelho.Estas restrições implicam em uma série de requisitos para os protocolos de comunicação nunca antes encontrados em tal escala.

3. Hardware Livre Arduino

O Arduino® faz parte do conceito de hardware e software livre, ou seja, está aberto para uso e contribuição de toda sociedade. O conceito deArduino surgiu na Itália em 2005, com o objetivo de se criar um dispositivo para controlar projetos/protótipos construídos de uma forma menos dispendiosa do que outros sistemas disponíveis até então no mercado.

Arduino é uma plataforma de computação física, baseada em uma simples placa de entrada/saída microcontrolada e desenvolvida sobre uma biblioteca que simplifica a escrita da programação em C/C++. O Arduino pode ser usado para desenvolver artefatos interativos stand-alone ou conectados ao computador através de Adobe Flash®, Processing®, Max/MSP®, Pure Data® ou SuperCollider®, a (figura 1) ilustra o arduino.

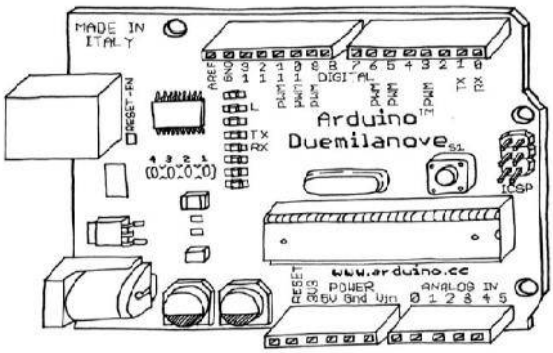


Figura 1. Arduino Duemilanove.

Um microcontrolador é um computador em um chip, que contem microprocessador, memória e periféricos de entrada/saída. O microprocessador, por sua vez, pode ser programado para funções especificas, em contraste com outros microprocessadores de propósito geral.Os microcontroladores utilizados no Arduino são da família Atmeal®, para nosso estudo utilizaremos o modelo Arduino Uno R3® que utiliza o microcontroladorATmega328P.

O núcleo AVR combina um rico conjunto de instruções com 32 registos de uso geral de trabalho. Todo os 32 registradores estão diretamente ligados à unidade lógica e aritmética (ULA), permitindo que os dois registrem independentemente ao ser acessado em uma única instrução executada em um ciclo de clock.A arquitetura resultante é código mais eficiente ao conseguir throughputs até dez vezes mais rápido do que os convencionais microcontroladores CISC.

4. Trabalhos Relacionados

O trabalho de Gomes (2012) apresenta um projeto e construção de um sistema de monitorização de energia baseado na utilização de um Arduino. O trabalho tem como objetivo realizar a monitoração de consumo de energia numa determinada instalação bem como monitorar alguns parâmetros relativos a qualidade de energia elétrica recorrente. O sistema permite o processamento de informaçõesem tempo real, a elaboração de gráficos e tem como principal objetivo ajudar o consumidor a realizar uma boa gestão energética.

Saldanha (2011) desenvolveu um sistema de redes de sensores sem fio para o monitoramento de equipamentos eletrônicos usando a plataforma de prototipação rápida Arduino e comunicação sem fio usando o protocolo de comunicação ZigBee, o trabalho apresenta também um algoritmo que visa maximizar o tempo de vida de uma rede e um modulo de RSSF, provendo menor consumo de energia e menor custo de desenvolvimento.

No trabalho de Trentin (2012) foi desenvolvido um sistema domótico simplificado baseado na plataforma arduino visando um maior relação custo benefício de seu desenvolvimento. O sistema permite a interação do usuário através da internet via web, ou pela rede pública de telefonia utilizando-se de processamentos digital de sinal com algoritmos de decodificação de sinais DTMF (Dual ToneMultiFrequency) que são gerados pelo telefone. O sistema se deu desde o desenvolvimento dos hardware necessários até o software que permitiu seu controle e comunicação com outros dispositivos. O projeto permite que seus usuários interagir com o sistema utilizando-se de um aparelho telefônico, ou através de dispositivos que permita acesso à internet.

Rodrigues e Assunção (2011) elaborarão um sistema de controle residencial centralizado em um *smartphone* dotado de um sistema operacional *android* que viabiliza a comunicação entre o dispositivo e o centralizador através de comunicação *bluetooth*, capaz de controlar todo o sistema de iluminação e o sistema de segurança residencial em uma única aplicação. O sistema é desenvolvido sobre a placa de circuito Arduino, um projeto de o*pen-source hardware* que facilita o desenvolvimento de aplicações físicas oferecendo funções de leitura e manipulação de elementos físicos.

O centralizador comunica-se diretamente com o smartphone utilizando bluetooth, não havendo assim a necessidade de um computador ou um roteador para intermedia a troca de informações entre os dispositivos controladores.

5. Arquitetura Proposta

De uma forma geral, a sistema foi projetado segundo a arquitetura ilustrada na (Figura2).

Figura 2: Arquitetura Proposta

Osensores de levantamentos dedados (sensores de corrente e sensores de tensão)são inicializados paralerossinaisdecorrente e tensão dos dispositivos monitorados, como lâmpadas, televisores, geladeira, central de ar, tomadas de tensão. Estes estão ligados diretamente no agente de controle, que controla o input/output desses dispositivos, através das requisições do agente administrador.Os dados lidos serão então condicionados de forma a ajusta-los, aos níveis padrões exigidos definidos no agente administrador para a prevenção da queima dos aparelhos.As informações coletadas sãomedidaspelo o agente administrador que por sua vez calculará a energia consumida e encaminha para um servidor em nuvem, todas as informações coletadas. Estes dados serão armazenados na base de dados,DB Coleta.

Devido a definição de um padrão de consumo o agente de anomalia,necessário para detectar padrões de consumo, utilizados para posteriormente identificar os diferentes níveis consumidos pelo usuário, fora desse padrão preestabelecido através de treinamento dos dispositivos sob condição normal de uso, definindo o DB de Análise. Os dados coletados pelos sensores são comparados com as informações de análise, caso

haja alguma anomalia, são disparados comandos de acordo com o DB de controle, para o gerenciamento do dispositivo.

6. Protótipo

De acordo com a forma de controle, ou seja, as formas como os diferentes elementos do sistema de controle estão distribuídos dentro da arquitetura do sistema, define-se a forma de controle como sendo centralizada ou descentralizada (ALIEVI, C. A).

Na arquitetura descentralizada existem diversos controladores interconectados por um *bus* que possibilitam o envio de informações entre eles. Já os atuadores, as interfaces e os sensores não necessariamente comunicam-se com mais de um controlador diretamente, ou seja, a proposta é dividir o sistema para suprir necessidades complexas (MARIOTONI, C. A. e ANDRADE Jr., E. P). Esse modelo de arquitetura tem como benefícios tornar os sistemas mais robustos a falhas, fácil desenho das instalações, grande facilidade de uso, ou seja, cumpre todos os requisitos que um sistema domótico deve ter.

A arquitetura do sistema proposto é utilizada tanto na forma centralizada quanto descentralizada pois faz o uso de sensores para viabilizar a troca de informação no sistema controlados e o usuário , dividida em duas partes que são o coordenador da rede e os dispositivos remotos, onde todos fazem uso do kits Arduino para coordena as tarefas previamente estabelecidas e *shield ethernet* para viabiliza a comunicação remota.

O dispositivo administrador da rede, que recebera as informações advindas dos sensores previamente instalados na rede para que se posas ter o conhecimento da corrente e tensão que circula pela rede e assim processar estas informações informando ao usuários seus detalhes sobre o estado dos do equipamentos. Os agentes de controle que é composta por uma placa relé senda a responsável por acionar ou desligar um aparelho.

Os relés funcionam controlando a circulaçãode uma corrente através de suas bobinas, que assim criara um campo magnético que atrairá um ou uma série de contatos fechando ou abrindo circuitos.(Figura3)ilustra o censor utilizado no projeto, e como visamos avaliar os sinais de corrente e de tenção que trafegam pelos aparelhos, será necessário monitorae condicionar os sinais de correntepara avaliar o comportamento de transdução eletromagnética passiva capaz de transformar uma amplitude de corrente em uma amplitude de tensão que, por sua vez, será aplicado à entrada analógica do microcontrolador. Logo, isto é necessário, pois o projeto trabalha com tensões de entrada de até 5 volts. Assim sendo, a tensão de saída estará em uma escala de -2,5 V a 2,5 V, de forma que se obtenha uma amostragem real do sinal elétrico em uma escala menor (LIMA, E. S).

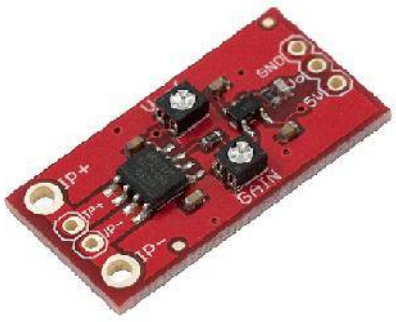


Figura3: Sensor de Corrente Ajustável ACS-712

O sinal de corrente, entregue pelo sensor, é proporcional à corrente primária, ou seja, também será alternada. O conversor A/D do Arduino trabalha com uma faixa dinâmica de tensões de entrada de 0 a 5V. Portanto foi necessário o condicionamento do sinal de corrente, convertido em tensão, se tomado nos terminais do resistor de carga, colocado em paralelo com a entrada do conversor A/D. O circuito do sensor de tensão utiliza um transformador 220VAC: 12VAC, o qual faz o isolamento galvânico dos circuitos de alta e baixa tensões, gerando um sinal de saída proporcional à tensão de entrada. Semelhante ao sinal de corrente, o sinal de tensão na saída estará na casa de -2,5V a 2,5V, necessitando também de condicionamento. A (Figura4) ilustra o circuito de condicionamento do sinal de tensão.

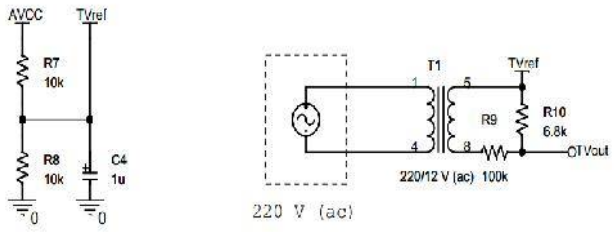


Figura4: protótipo do circuito referente ao condicionamento dos sinais de tensão

Na(Figura 5) ilustramos o protótipo a ser montado para que se possam adicionar os aparelhos e verificar seu gasto energéticos.

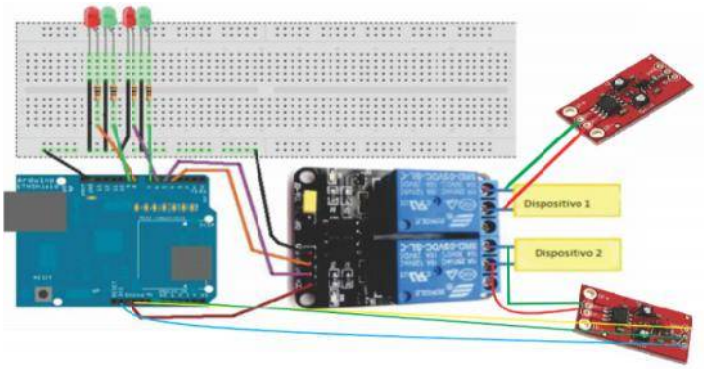


Figura 5: Protótipo

A aplicação consiste em um Arduino que será o responsável por centralizar todas as tarefas e será através dele que ao receber comandos do usuário, realizara disparos para que se possa estar liberando a passagem de corrente elétrica e acionar os aparelhos conectados aos relés *shield* que recebera os equipamentos elétricos da rede, em cada par de relé será adicionado um sensor que serão os responsáveis por realizarem a leitura da corrente elétrica. No modelo de *shield* acima podemos ligar 2 equipamentos. Para ligar nosso equipamentos ligaremos um dos fio de tensão azul 110v ou 220v no primeiro conector do rele que é o de posição 'normalmente aberto' que fica do lado das numerações e outro fio vermelho ligaremos no pino do meio que é o 'comum' faremos isso para cada equipamento que quisermos adicionar e em seguida conecta-se ao neutro da rede elétrica.

Um circuito é montado sobre um *protobooard* que consiste em uma placa didática composta de uma matriz de contatos que permite a construção de circuitos experimentais sem a necessidade de efetuar a solda dos componentes, isso permitiu que fosse efetuada

uma série de experimentos com os mesmos componentes inserindo ou removendo os mesmos com rapidez e segurança. Para auxiliar o usuário a identificação do aparelho em uso no momento, será realizado o acionamento de *leds (Light emitterdiode)* que são componente eletrônico semicondutor, ou seja, um diodo emissor de luz, que se utiliza da mesma tecnologia utilizada nos chips dos computadores, e tem a propriedade de transformar energia elétrica em luz, indicando o estado momentâneo do relé, onde o mesmo pode-se encontrar-se com a presença de corrente elétrica ou ausência. Na (Figura 6) demonstramos como o sistema funciona, ilustrando o funcionamento de todo o sistema.

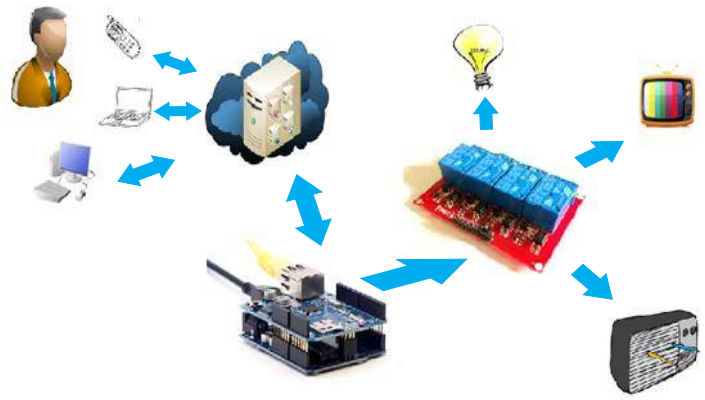


Figura 6: Funcionamento do Protótipo.

A aplicação foi desenvolvido em linguagem C e compilado pelo próprio ambiente de desenvolvimento do Arduino. A aplicação foi criado para adquirir amostras dos sinais provindos dos sensores, efetuar o cálculo do consumo, a potência ativa, a potência aparente, o fator de potência e assim apresentar os resultados desses cálculos na aplicação desenvolvida para que seja realizado o monitoramento remotamente.

7. Teste

O protótipo ficou em funcionamento por 30 dias ininterruptos para que assim pudesse ser realizadas avaliações sobre o seu funcionamento e constatar se os seus objetivos inicialmente pretendidos estavam sendo compridos. Para isso foi desenvolvido uma página web para que assim fosse possível administra os dispositivos através de trocas de informações com o sistema.Coletou-se um total de 170mb de informações referentes ao consumo do usuário dando valores das potencias ativas, potencias aparentes, fator de potência, tensão e corrente.

O sistema se comportou de forma bem satisfatória onde para a realização do teste foi adicionado apenas um aparelhos para testar as funcionalidades do sistema e assim podemos coletar as informações descrita na seção dos resultados obtidos.

8. Resultados E Discussão

Os testes realizados para a avaliação da aplicação teve como base apenas cargas resistivas, de modo que fosse possível simplificar a observação do funcionamento dos resultados obtidos, permitindo facilmente a realização de ajustes e possíveis alterações sem que fosse comprometida a interpretação dos valores obtidos para este experimento. Por obter no momento dos teste apenas um sensor o experimento foi desenvolvido apenas com uma Lâmpada 100W/220V.

Para validar o funcionamento do medidor, foram utilizados equipamentos como voltímetros e amperímetros a fim de realizar os mesmos experimentos e assim poder comparar os valores obtidos com os valores fornecidos pela aplicação desenvolvida. As amostras fornecidas pela aplicação foram processadas pelo software desenvolvido e processados no Arduino e entregues via comunicação serial pela saída USB do microcontrolador. Os dados da medição feitas foram coletados e registrado no banco de dadosusado para que esta informações fossem usadasposteriormente caso seja necessário.

O objetivo desses testes foi monitorar as o desperdício de energia em um ambiente residencial e assim propor uma melhor utilização no consumo e assim poder observar as características de funcionamento do protótipo e da própria carga que foi monitorada. Os resultados provenientes deste experimento podem ser vistos nas (Tabelas 1).

Tabela 1 – Dados relativos à medição com carga Lâmpada Incandescente.

Carga Potência ativa (W)	Potência aparente (VA)	Fator de potência (%)	Tensão RMS (V)	Corrente RMS (A)	Consumo (watts/hora)
981,02	1245,42	78,77	214,76	5,8	18
976,04	1198,13	81,46	214,66	5,58	18,31
975,95	1216,49	80,23	214,84	5,66	18,62
971,27	1205,75	80,55	214,67	5,62	18,92
969,80	1212,47	79,98	215,08	5,64	19,23
966,95	1157,09	83,57	214,88	5,38	19,53
996,29	1255,34	79,37	215,11	5,84	19,84
979,09	1232,63	79,43	214,91	5,74	20,15
985,13	1275,79	77,22	215,26	5,93	22,92
959,76	1115,99	86	215,06	5,19	23,22
976,73	1253,91	77,9	215,25	5,83	23,53
971,54	1247,99	77,85	215	5,8	23,83
964,87	1186,7	81,31	215,16	5,52	24,14
977,32	1220,32	80,09	214,93	5,68	24,44
986,2	1268,57	77,74	215,11	5,9	24,75
974,14	1212,43	80,35	214,83	5,64	25,06
972,98	1201,05	81,01	214,87	5,59	25,37
989,45	1327,46	74,54	214,63	6,18	25,68

9. Considerações Finais

Neste trabalho foi apresentado um projeto de baixo custo, voltado à sistemas de automação residencial. Em seu desenvolvimento buscou-se implementar um sistema que fosse simples e o mais viável possível para que o mesmo pode-se ser utilizado por um maior número de usuários, logo, teve-se o cuidado de fazer uso de componentes fáceis de serem achados no mercado. O diferencial desse sistema é que pode-se adequar-se a qualquer tipo de pessoa, levando em consideração o seu manuseio simplificado.

O projeto disponibiliza uma maior comodidade e praticidade para os usuários em permiti o monitoramento remoto, podendo assim diminuir o desperdício sobre o consumo de energia com uma maior eficiência.

10. Referencia

LIMA, E. S. **Protótipo de tarifado digital de energia elétrica. (**2007) Monografia (Graduação em Engenharia) – Faculdade de Engenharia – Pontífice Universidade Católica de Goiás, Goiânia, 2007.
ALIEVI, C. A. **Automação residencial com utilização de controlador lógico(**2008) Trabalho de Conclusão de Curso (Especialização) – Instituto de Ciências Exatas e Tecnológicas, Centro Universitário FEEVALE, Novo Hamburgo, 2008.

AURESIDE. **Associação Brasileira de Automação Residencial(**2012) Disponível em: http://www.aureside.org.br. Acessadoem: 01 jan. 2013.

BOLZANI, (2007), Caio Augustus Morais, **Residências inteligentes**. São Paulo, Ed. Editora e Livraria da Física.

BRUGNERA, Mauro Ricardo.**Domótica**(2008). Disponível em: <http://www.unibratec.com.br/jornadacientifica/diretorio/FEEVALE+MRB.pdf>. Acesso em: 02 nov. 2012.
ANGEL, P. M. **Introducciónala domótica; Domótica: controle e automação**. (1993) EscuelaBrasileño-Argentina de Informática. EBAI.

DIAS, C.L.A.; PIZZOLATO, N.D. **Domótica: Aplicabilidade e Sistemas de Automação Residencial**, (2004) CEFET. Campos dos Goytacapes - RJ.

EUZÉBIO, M.V.M.; MELLO, E.R. **DroidLar: Automação Residencial através de um celular Android**.(2011) IF-SC.

SANTOS, P.D.B; **Sistema de Monitoramento de Energia Elétrica**. (2012) FCT - Universidade Nova de Lisboa.

LINS, Vitor. E MOURA, Waldson. **Domótica: Automação Residencial**, Disponível em http://www.unibratec.edu.br/revistacientifica/n5_artigos/lins_moura.pdf, acessado em 08/02/2013.

Loureiro at al, **Redes de Sensores Sem Fio,** XXI Simposio Brasileiro de Redes de Computadores, Disponivel em http://www.sensornet.dcc.ufmg.br/publica/pdf/179_Loureiro_Nogueira_Ruiz_Mini_Nakamura_Figueiredo.pdf, acessado em 08/02/2013.

MASSIMO, Banzi. E-book - **GettingStartedwith Arduino**, (2008).

MCROBERTS, Michael. (Tradução Rafael Zanolli), **Arduino Básico**, (2011) Editora Novatec.

MARIOTONI, C. A. e ANDRADE Jr., E. P., **Descrição de Sistemas de Automação Predial Baseados em Protocolos PLC Utilizados em Edifícios de Pequeno Porte e**

Residências (2002), Revista de Automação e Tecnologia de Informação. Volume 1, número 1.

RODRIGUES, E. A E OLIVEIRA, P. R.V, **Controle de Automação Residencial Utilizando Celulares com Bluetooth**, (2011) UNIVALI.

SALDANHA, I. A. **Redes de Sensores sem Fio para Monitoramento de Equipamentos Eletrônicos**. (2011). Dissertação (Mestrado em Informática) – Pontifícia Universidade Católica de Minas Gerais, Belo Horizonte, 2011.

TRENTIN, P. M. **Domótica via Dispositivos Móveis com Arduino**. (2012). Monografia (Graduação em Ciência da Computação) – Universidade do Oeste de Santa Catarina, Videira, 2012.

Uma Proposta Para Controle De Densidade em Redes de Sensores Sem Fio utilizando Inteligência Computacional

Diego Carvalho P. Macedo[1], Harilton Da S. Araújo[2], Aldir S. Sousa[3]

[1]Centro de Ensino Unificado de Teresina - CEUT
Av. dos Expedicionários, 790 – Bairro São João
Caixa Postal 64.046-700 Teresina, PI

diegomacedo_cp@hotmail.com, {hariltonarujo,aldirsousa}@ceut.com.br

Abstract. *The wireless sensor networks (WSN) consist of small sensors capable of capturing information phenomena of the environment they are inserted and to transmit this information. The sensor nodes are very limited in their processing power and capacity and energy storage. When it has a fairly large number of nodes in an environment, energy consumption is greater. This paper presents a proposal for density control in wireless sensor networks using computational intelligence. The simulations indicate that the proposal is effective with respect to reduction of energy consumption, increase in the number of messages sent and received, increased the residual energy of the network, increasing the lifetime of the network and reducing the cost of receiving the message.*

Resumo. *As redes de sensores sem fio (RSSF) são compostas por pequenos sensores capazes de captar informações de fenômenos do meio ambiente em que estão inseridos e de transmitir essas informações. Os nós sensores são bastante limitados em seu poder de processamento e a capacidade de energia e de armazenamento. Quando se possui um número bastante elevado de nós em um ambiente, consumo de energia é maior ainda. Este trabalho apresenta uma proposta para controle de densidade em redes de sensores sem fio utilizando inteligência computacional. As simulações indicam que a proposta é eficiente no que diz respeito a redução do consumo de energia, aumento no número de mensagens enviadas e recebidas, aumento na energia residual da rede, aumento na vida útil da rede e a redução do custo de recebimento da mensagem.*

1. Introdução

Uma rede de sensores sem fio (RSSF) pode ser definida como um conjunto de nós sensores que têm a função de captar as ações do meio em que estão inseridos e, por meio de comunicação sem fio, podem transmitir informações para outros nós da rede. Estas redes diferem das redes de computadores tradicionais em diversos aspectos. Geralmente, as RSSF têm um grande número de nós sensores distribuídos, possuem restrições de energia e processamento e devem ter mecanismos para autoconfiguração em caso de perda de comunicação e falhas nos nós sensores (SOBRAL et. Al, 2013).

Um dos principais desafios na área de RSSF é a redução do custo de energia nos nós sensores, uma vez que na maioria dos casos, estes nós estão localizados em locais

de difícil acesso, o que torna inviável a substituição periódica das baterias que os alimentam.

Dada a característica de baixa alimentação energética dos nós sensores, algoritmos de roteamento devem visar o consumo mínimo de energia. Neste trabalho, propõe-se uma nova abordagem para lidar com roteamento em RSSF. A abordagem aqui proposta destaca-se pela aplicação de um sistema de inferência *fuzzy* ao algoritmo de roteamento *Inter Cluster Routing Algorithm* (ICA) (HABIB, 2004).

Com objetivo de analisar o real ganho de desempenho da abordagem proposta neste trabalho, levaram-se em consideração algumas métricas que indicam o melhor desempenho da rede. As métricas utilizadas neste trabalho foram: número de mensagens enviadas, mensagens recebidas, energia residual, custo de recebimento de mensagens e o tempo de vida da rede.

Uma forma de melhorar o desempenho de uma RSSF pode ser por meio de um algoritmo de roteamento eficiente. Este trabalho consiste em aperfeiçoar o algoritmo de roteamento ICA para lidar com o roteamento visando melhorar o desempenho da RSSF. Neste trabalho, propõe-se a aplicação de um sistema de inferência *fuzzy* no algoritmo de roteamento ICA no intuito de se monitorar um evento em um espaço geográfico, no objetivando otimizar as rotas para maximizar o desempenho da rede. Com a abordagem proposta neste artigo, o sistema *fuzzy* possibilita que a rede se autoconfigure no intuito de atender regiões geográficas em que ainda haja energia suficiente e que esteja próxima a algum fenômeno.

2. A Metodologia Proposta

Redes de Sensores sem Fio (RSSF) é uma funcionalidade que promete monitorar, instrumentar, e, possivelmente, controlar o meio físico. Estas redes são constituídas de um número elevado de dispositivos sem fios (nós sensores ou simplesmente sensores). Esses são distribuídos densamente por toda uma região de interesse (ALFREDO, 2006)**.**

A RSSF pode ser formada por centenas ou até milhares de sensores posicionados dentro do fenômeno a ser observado ou próximo a ele, os quais são, de fato, dispositivos compostos de transceptor, fonte de energia, unidade de sensoriamento, processador e memória. Como as redes de sensores sem fio são capazes de se auto organizar, torna-se dispensável um planejamento minucioso de posicionamento dos sensores.

Cada um dos sensores é alimentado por uma fonte de energia escassa, ou seja, tem curta duração. Portanto, o consumo de energia influencia a durabilidade da RSSF. Sabendo disso, faz-se mister a aplicação de algoritmos de roteamento eficientes, que minimizem a transmissão de informações desnecessárias e/ou redundantes. Este trabalho propõe uma nova abordagem no intuito de minimizar o envio e recepção de informações ao aplicar mecanismos de inteligência computacional para aperfeiçoar um algoritmo de roteamento clássico da literatura.

Nas próximas seções destacam-se o algoritmo de roteamento utilizado neste trabalho e o sistema de inferência *fuzzy* aplicado para seu aperfeiçoamento.

2.1. Algoritmo de Roteamento ICA

O Inter Cluster Routing Algorithm (ICA) é um algoritmo de roteamento em RSSF baseado no algoritmo de roteamento LEACH (HABIB, 2004). No ICA, quando há o início do funcionamento da rede, a estação base envia mensagem (*broadcast*) para os nós da rede informando sua posição geográfica do ambiente. Após isso, os nodos sabem a posição geográfica do *sink*, suas devidas posições e a distância e a rota até o *sink*. O ICA possibilita ao nó se tornar líder (*cluster head*) de uma determinada região. Depois disso, os nós próximos se conectam a *cluster head* e os nós que possuírem informações a serem enviadas para o *sink* mandará para o *cluster head*, para depois esse repassá-las para a estação base. Depois de um nó se eleger líder, esse manda um *broadcast* par os nós próximos avisando-os de que ele se tornou um *cluster head*. Assim, estes nós terão a posição geográfica deste nó líder e terão uma rota mais curta até ele. Com isso, os nós sabem a localização do nodo líder mais próximo, com o qual devem se comunicar. Portanto, no ICA, os nós sensores não enviam as mensagens diretamente para o *sink*, e sim para outro *cluster head* mais próximo e que esteja na direção da estação base.

O algoritmo ICA possibilita uma degradação suave da energia da rede como um todo e não necessariamente nodo por nodo.

2.2. Utilização do *Fuzzy*

O sistema de interferência *fuzzy* foi utilizado no ICA de forma a selecionar os nós lideres que possuam a distância mais próxima a um local que se esta acontecendo algum tipo de fenômeno no ambiente e que possua energia suficiente, portanto o sistema exerce a função de identificar aqueles nós que podem se tornarem lideres, por exemplo, quando se acontece um evento em algum local, a rede se auto configura de forma com que os nós determinem suas distâncias ao evento mais próximo, após isso entra o funcionamento do *fuzzy* que com o dado da distância e a energia do nó, elegera os nós permitidos na rede, na próxima seção será apresentado à característica do sistema *fuzzy*.

2.3. Sistema de Interferência *Fuzzy*

Um sistema de interferência *fuzzy* foi aplicado e tem fundamental importância para o bom desempenho desta proposta. O sistema de inferência *fuzzy* aplicado neste trabalho é baseado em regras linguísticas do tipo **se** <condição> **então** <ação>. O sistema de inferência *fuzzy* aplicado neste trabalho é composto por uma interface de *fuzzyficação*, uma base de regras o procedimento de inferência e a interface de *defuzzificação*, conforme a Fig. 1 (SOUSA e ASADA, 2011).

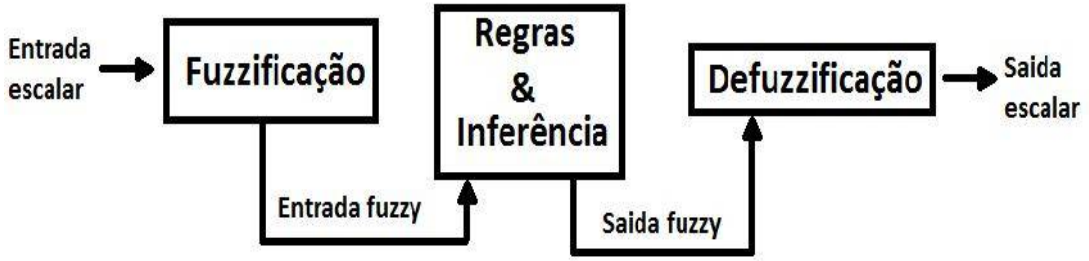


Fig. 1 Estrutura Básica de um Controlador *Fuzzy*

- *Interface de fuzzyficação*: é o processo de mapear valores escalares para graus de pertinência aos termos linguísticos dos conjuntos *fuzzy*.

- *Procedimento de inferência*: consiste em mapear valores de entra para saídas *fuzzy* de acordo com as regras *fuzzy*. Portanto diversas regras *fuzzy* são combinadas para produzir a saída do sistema de inferência.

- *Interface de defuzzyficação:* consiste em converter as saídas *fuzzy* em valores escalares.

2.3.1. Funções de pertinência e conjunto de regras *fuzzy*

O sistema *fuzzy* desenvolvido neste artigo baseia-se em duas variáveis de entrada e uma de saída. As variáveis *fuzzy* de entrada são *distância do nó* e *energia contida no nó*. A variável de saída, aqui chamada de *chance de se tornar líder*, consiste em saber se o nó tem chance ou não para se tornar líder no ICA.

Os conjuntos *fuzzy* atribuídos à variável *distância do nó* foram: Longe (L), Distância Média (DM) e Perto (P). O universo de discurso destes conjuntos está no intervalo discreto entre 0 a 5. Para a *energia contida no nó,* foram atribuídos seguintes conjuntos: Muito Baixa (MB), Baixa (B), Média (M), Alta (A) e Muito Alta (MA). O universo de discurso destes conjuntos está no intervalo contínuo entre 0 e 5. A *chance de se tornar líder* é composta pelos seguintes conjuntos *fuzzy*: Pouca Chance (PC), Chance Média (CM) e Muita Chance (MC). O universo de discurso destes conjuntos está no intervalo contínuo entre 0 e 30.

Todas as variáveis de entrada foram modeladas como função de pertinência trapezoidal. A Eq. (1) define a função trapezoidal para os parâmetros *a,b,c,d*.

$$f(x) = \begin{cases} 0, & if\ x \leq a \\ \dfrac{x-a}{b-a}, & if\ a < x \leq b \\ 1, & if\ b < x \leq c \\ \dfrac{d-x}{d-c}, & if\ c < x \leq d \\ 0, & if\ x > d \end{cases} \qquad (1)$$

A seguir, listam-se os parâmetros TRAPE[a b c d] para cada um dos conjuntos *fuzzy* utilizados neste trabalho:

- L: TRAPE[3 4 5 5]
- DM: TRAPE[1 2 3 4]
- P: TRAPE[0 0 1 2]
- MB: TRAPE[0 0 0.5 2]
- B: TRAPE[0.5 1 1.5 2]
- M: TRAPE[1.5 2 2.5 3]
- A: TRAPE[2.5 3 3.5 4]

- MA: TRAPE[3.5 4 5 5]

Na Tabela 1, resumem-se as regras linguísticas do *fuzzy* utilizada neste artigo. Os resultados serão PC (Pouca Chance), CM (Chance Média) e MC (Muita Chance):

Tabela 1: Regras linguísticas

		Energia				
		MB	**B**	**M**	**A**	**MA**
Distância	**L**	PC	PC	PC	PC	PC
	DM	PC	PC	CM	CM	MC
	P	PC	PC	CM	MC	MC

3. Testes computacionais

Com objetivo de avaliar o desempenho da abordagem proposta neste trabalho simulações foram realizadas através do simulador Sinalgo (SINALGO, 2013). Este simulador foi escolhido por ser largamente utilizado na literatura e de fácil configuração.

O pacote java jFuzzyLogic (jFUZZYLOGIC, 2013) foi utilizado para aplicar o sistema de inferência *fuzzy*.

3.1 Ambiente dos testes

As simulações foram realizadas em um Computador *HP Pavilion dv4 – 2040br*, com sistema operacional Windows 7, processador *Intel Core i3 CPU M 330 @ 2,13GHz, 2128MHz*. Memória RAM de 4 GB DDR3 e memória cache L2 256 KB.

3.2 Características da rede

As simulações foram realizadas em redes simuladas compostas de 100 nós sensores e um *sink node,* dispostos de forma aleatória em um ambiente de duas dimensões sendo de 1000x1000. O tempo em *rounds* determinado para cada simulação foi de 8.000 *rounds* e as posições geográficas dos nós foram salvas para que todas as simulações fossem realizadas com a mesma configuração, dando segurança às comparações realizadas. Foi configurado para que cada nó sensor possua uma bateria com capacidade inicial de 5 J (joule).

Nas simulações realizadas foi utilizado o modelo de dissipação de energia proposto por Heinzelman et al. (2003), por ser o amplamente aceito e utilizado na literatura. O modelo de Heinzelman et al. (2003) pode ser sintetizado conforme segue.

Energia gasta na transmissão:

$$E_{Tx}(k, d) = E_{elec} * k + E_{amp} * k * d^2$$

Energia gasta na recepção:

$$E_{Rx}(k) = E_{elec} * k$$

Especificações:

k = número de bits da mensagemd = distância

E_{Tx} = Energia gasta na transmissão de mensagem

E_{Rx} = Energia gasta na recepção da mensagem

E_{amp} = Energia do amplificado de transmissão

Neste trabalho, E_{elec} = *50 nJ/bit e* E_{amp} = *100 pJ/bit/m2*. Nas simulações realizadas, o tamanho das mensagens que trafegam na rede foi configurado para 500 *bytes*, que é igual a 4000 bits.

Na unidade de processamento, a energia dissipada nas execuções dos protocolos de roteamento e no processamento.

No modelo de comunicação, a energia gasta no processamento equivale à energia dissipada na transmissão de um bit a 100 metros dividido por 3.000, conforme a Eq. (2).

$$E_{Trans} = E_{Cir} * 1 + E_{Amp} * 1 * 100^2 \qquad (2)$$

Logo:

$$(E_{Cir} + 10000 * E_{Amp})/3000 \qquad (3)$$

onde

ETrans = Energia de transmissão.

Eelec = ECir = Energia gasta nos circuitos do rádio.

EAmp = Energia do amplificador de transmissão.

Neste trabalho, foram realizadas 10 simulações para cada cenário a fim de obter dados, para fazermos a comparação destas informações.

3.3 Resultados

Os resultados exibem a comparação de dois cenários: no primeiro foi utilizado o algoritmo de roteamento ICA clássico. No segundo cenário, foi utilizado o algoritmo de roteamento ICA com a utilização do sistema de inferência *fuzzy* no intuito de obter o controle de densidade dos nós na rede.

As métricas utilizadas como forma de comparação dos dois cenários foram: a quantidade de mensagens enviadas, a quantidade de mensagens recebidas, a energia restante da rede, o custo de recebimento de mensagens e o tempo de vida da rede. Abaixo, definem-se as métricas utilizadas.

- A quantidade de mensagens enviadas se refere a quantas mensagens a rede dos sensores tentaram mandar para estação base;
- A quantidade de mensagens recebida se refere à quantidade de mensagens que o *sink* conseguiu receber dos nós.
- A métrica da energia restante da rede se refere ao somatório das energias restantes em cada nó da rede nó fim de cada simulação.
- A métrica do custo do recebimento de mensagens leva em consideração a quantidade de mensagens recebidas pela energia residual total da rede, obtendo assim, o valor do custo do recebimento da mensagem.

A métrica do tempo de vida da rede se refere até quando a estação base conseguiu receber mensagens. Assim que a estação base deixar de receber mensagens a rede será considerado como morta.

A Fig. 2 mostra os resultados das simulações tendo como métrica a quantidade de mensagem enviada ao *sink*.

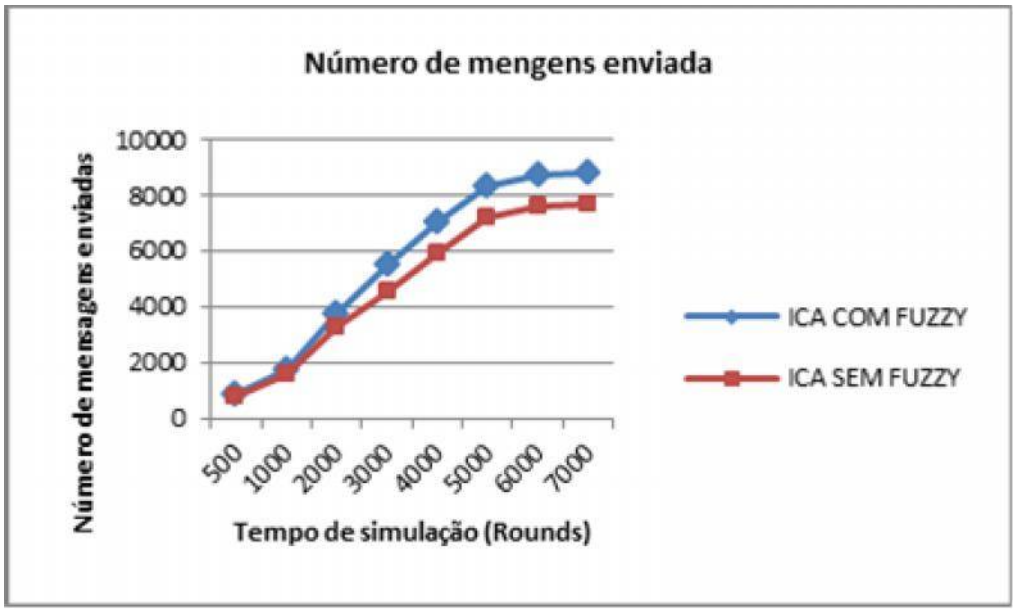


Fig. 2 Quantidade de mensagens enviadas x rounds (tempo)

O cenário do ICA com a utilização do *fuzzy* obteve os resultado significativamente melhores do que o cenário do ICA clássico. O ICA com *fuzzy* começa a se destacar no número de envios de mensagens pelos nós sensores ao *sink* a partir de 1.000 rounds (Fig. 2).

A Fig. 3 mostra os resultados das simulações tendo como métrica a quantidade de mensagem recebida no *sink*.

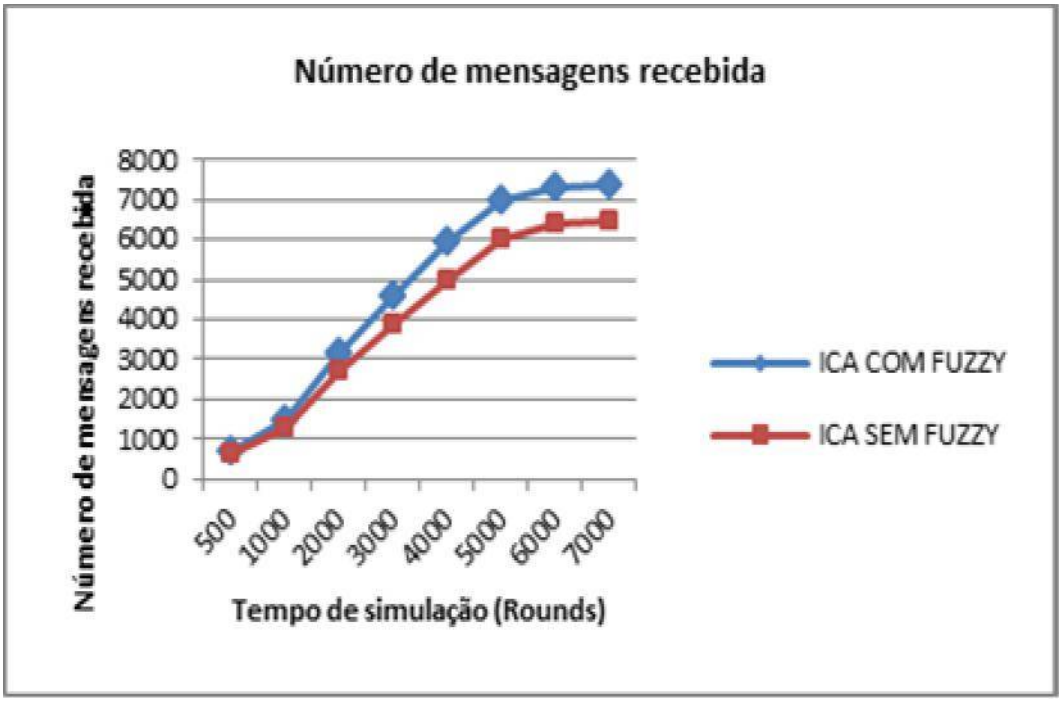


Fig. 3 Quantidade de mensagens recebidas x rounds (tempo)

O cenário do ICA com a utilização do *fuzzy* teve melhor desempenho também para a métrica quantidade de mensagens recebidas por *rounds*. O ICA com *fuzzy* começa a se destacar no número de envios de mensagens pelos nós sensores ao *sink* a partir dos 1000 rounds (Fig. 3).

A Fig. 4 mostra os resultados das simulações tendo como métrica a energia residual da rede. Como se pode observar na referida figura, o ICA com *fuzzy* obteve um melhor consumo de energia na rede, fazendo com que a rede não desperdiçasse energia e obtendo assim uma maior economia de energia.

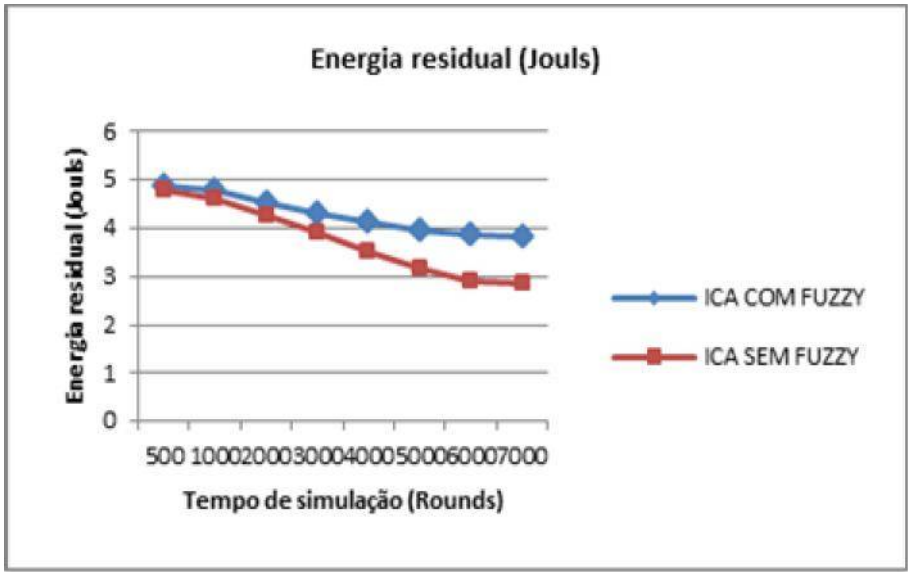


Fig. 4 Energia residual da rede x rounds (tempo)

A Fig. 5 mostra os resultados das simulações tendo como métrica de comparação o custo de recebimento de mensagens da rede, que é a razão da quantidade de mensagens recebida pela energia residual total da rede.

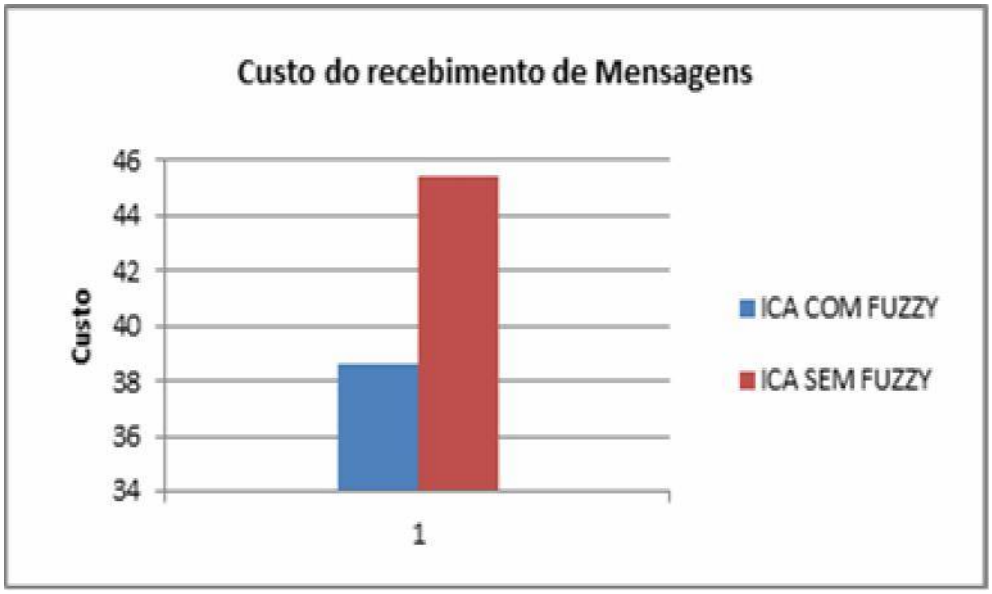


Fig. 5 Custo do recebimento de mensagens

O ICA clássico possui um valor bastante elevado no custo de recebimento de mensagens, sendo, portanto o ICA com *fuzzy* muito mais viável do que o ICA clássico (Fig. 5).

A Fig. 6 mostra os resultados das simulações tendo como referência o tempo de vida útil da rede, que é o tempo necessário para que a rede pare de funcionar.

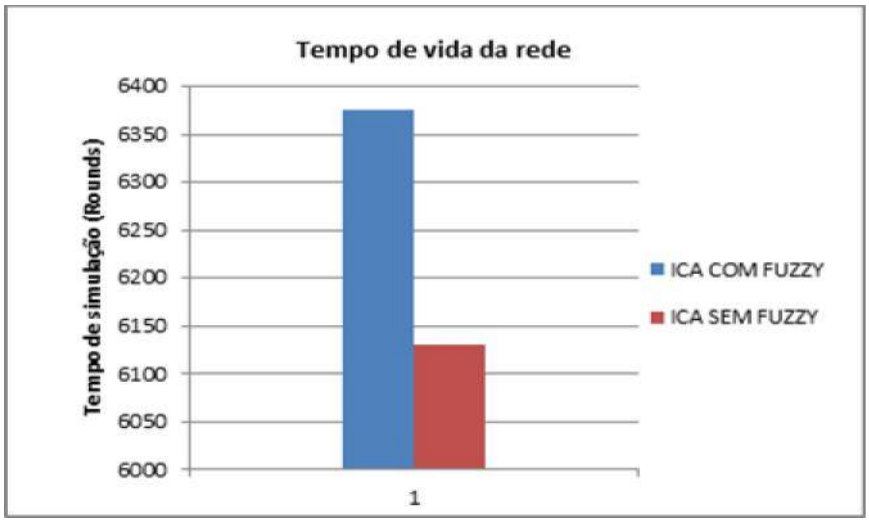


Fig. 6 Tempo de vida da rede

Em relação ao tempo de vida da rede, o ICA com *fuzzy* se destaca do ICA clássico de forma acintosa. Ao final de 8.000 rounds, o algoritmo ICA com *fuzzy* foi 305% melhor (Fig. 6).

4. Conclusão

Neste trabalho propõe-se uma nova metodologia para lidar com roteamento em redes de sensores sem fio. A abordagem proposta neste trabalho consiste na aplicação de um sistema de inferência *fuzzy* para seleção dos nós líderes do algoritmo ICA clássico. Pôde-se verificar uma melhoria significativa no algoritmo ICA em diversas simulações realizadas.

O ganho de desempenho global da rede deve-se ao sistema *fuzzy*. Este sistema possibilitou que a rede se autoconfigurasse no intuito de melhorar o controle de densidade de nós em uma RSSF. Através da nova abordagem proposta neste trabalho, uma RSSF pode se autoconfigurar uma vez ocorrendo um evento futuro em outra região geográfica se a rede ainda possuir energia. O algoritmo ICA clássico não é capaz disso, já que os lideres são eleitos aleatoriamente. Portanto, nem sempre em locais apropriados. Isso incorre em maior gasto de energia.

Foram realizadas análises comparativas considerando o algoritmo ICA com o sistema de inferência *fuzzy* e o algoritmo ICA clássico. Levaram-se em consideração as seguintes métricas: número de mensagens enviadas, mensagens recebidas, energia residual, custo de recebimento de mensagens e o tempo de vida da rede. Em todos os testes realizados pode-se perceber um ganho significativo em desempenho da rede para todas as métricas analisadas.

Referências Bibliográficas

ALFREDO, Antonio. **Redes de Sensores Sem Fio.** Departamento de Ciência da Computação da Universidade Federal de Minas Gerais. Belo Horizonte. 2006.

DALTON, Giulian. **Roteamento em Redes de Sensores.** Instituto de Matemática e Estatística. Universidade de São Paulo. 2004

FONTENELE, Kérllon. **Uma proposta de modelo de dissipação de energia em redes sensores sem fio.** 2010. Teresina

HABIB, E., Câmara, D., and Loureiro, A. A. (2004). Ica: **Um novo algoritmo de**

roteamento para redes de sensores. In Simpósio Brasileiro de Redes de Computadores, Gramado, RS.

POTTIE, G. J. and KAISER, W. J. (2000). **Wireless integrated network sensors. Communications of the ACM**, 43(5):51-58.

RIBEIRO, Wellington Gomes, ARROYO, José Elias Claudio, SANTOS, André Gustavo, ROCHA, Mauro Nacif. **Aplicação da Meta-Heurística ILS para o problema de Cobertura Conectividade e Roteamento em RSSF 2D Plana homogênea**. In: ENCONTRO DE COMPUTAÇÃO E INFORMÁTICA DO TOCANTINS, 13., 2011, Palmas. **Anais...** Palmas: CEULP/ULBRA, 2011. p. 109-118.

SOUSA, A. S. e ASADA, E. N. **Combined heuristic with fuzzy system to transmission system expansion planning.** Electric Power Systems Research , Volume 81 (1) Elsevier, 2011.

SINALGO, **Simulator for Network Algorithms**. Disponível em: http://www.disco.ethz.ch/projects/sinalgo/. Acesso em 16/05/2013.

jFUZZYLOGIC. **Open Source Fuzzy Logic library and FCL language implementation**. Disponível em: http://jfuzzylogic.sourceforge.net/html/index.html. Acesso em 16/05/2013.

SOBRAL, J., SOUSA, A. S., ARAUJO, H., BALUZ, R., FILHO, R., LEMOS, M., e RABELO, R. **A Fuzzy Inference System for Increasing of Survivability and Efficiency in Wireless Sensor Networks**. In *ICN 2013, The Twelfth International Conference on Networks* (pp. 34-41).

Projeto Controle: Prototype of Telerobotic Using Python and Arduino

Marlo Z. B. Araujo[1]

[1]Universidade Estadual do Piauí (UESPI)
Av. Nossa Senhora de Fátima, S/N, Bairro de Fátima – 64202-220 – Parnaíba – PI – Brazil

zenibdea@hotmail.com

Abstract. *This article describes a solution for Telerobotic systems. Aims to show a solution to problems like exploring environments harmful to humans, providing the execution of tasks where the life of the researcher do not take risks. For this, we turned to the use of Python that will work together with a free platform circuit board, the Arduino which resulted in an architecture design of a robot in the form of a motor vehicle with four wheels. We found some problems that are presented in this work, as well as their possible solutions.*

Resumo. *Este artigo descreve uma solução para sistemas de TeleRobótica. Tem como objetivo mostrar uma solução para problemas como explorar ambientes nocivos para os seres humanos, proporcionando a execução de tarefas onde a vida do pesquisador não corra riscos. Para isso, vamos dirigiu-se ao uso da linguagem Python que irá trabalhar juntamente com uma placa de circuitos de plataforma livre, o Arduino que resultou em um projeto de arquitetura de um robô em forma de um veículo motorizado com quatro rodas. Foram encontrados alguns problemas que são apresentados neste trabalho, bem como suas possíveis soluções.*

1. Introdução

O presente trabalho apresenta um protótipo para Telerobótica intitulado Projeto Controle que surgiu da curiosidade de construir um protótipo de Telerobótica desenvolvido a partir do Python e Arduino. Essa temática causou a seguinte problemática: ausência de fontes teóricas que explanassem tal assunto. Baseado no exposto, afirma-se que se tornou desafiador saber como funcionava um protótipo de Telerobótica, usando as áreas de conhecimento supracitadas. Neste estudo, portanto, é sugerido um Protótipo de Telerobótica, usando o Python e Arduino, que realiza a comunicação entre dois ou mais computadores através da rede de forma mais simples, usando Socket, bem como transmite imagens pela rede, obtendo comunicação serial entre o Python e o Arduino.

Como aporte teórico, pautou-se na Telerobótica, que dentre outras funções, promove por meio de projetos, a exploração de grandes profundidades de fendas no fundo no oceano, superfície de outros planetas, desarmamento de bombas, manuseio de produtos radioativos, isso sem que haja a presença humana nestes ambientes nocivos.

Sob essa perspectiva a Telerobótica apresenta-se como a ciência que alia a Robótica com a Telemática sendo, ainda, uma área do conhecimento humano que reúne um conjunto e o produto da adequada combinação das tecnologias associadas à eletrônica, informática e telecomunicações, ou seja, a união da Telemática com a Robótica engendra o ato de controlar um robô à distância, integrando assim, várias áreas do conhecimento citadas por Pinheiro. A Telerobótica nesse sentido apresenta uma importância ímpar na medida em que, possibilita às pessoas a realização de ações a longas distâncias. O acesso de pesquisadores a ambientes hostis que podem oferecer algum risco a suas vidas, são solucionados a partir de equipamentos desenvolvidos para tal finalidade. Para realizar essa tarefa, é primordial um equipamento Telerobótico, principalmente em estudos onde são necessárias a resistência e precisão de um robô e o conhecimento de um especialista humano, promovendo a redução de risco de vida dos pesquisadores, uma vez que não seja necessário o operador do sistema e a máquina operada estarem no mesmo local.

Desde seu início modesto em 1940, quando o primeiro Tele Operador foi projetado, o foco foi principalmente para aplicações a serem realizadas no espaço, materiais nucleares e operações subaquáticas até os anos 80 [...] Tele Cirurgia, Telerobótica semi-autônoma, manutenção de linha viva de energia, e outros. Sua aplicação é diversificada, fornece a oportunidade de aumentar o conhecimento humano com explorações no espaço onde astronautas podem utilizar-se da Telerobótica para explorar outros planetas a partir de suas atmosferas com redução de custos e risco às vidas dos exploradores. Do mesmo modo em que cientistas e engenheiros alcançam profundezas dos mares remotamente a partir de navios na superfície. Em suma, escolheu-se a Telerobótica como referencial teórico, primeiro, pela afinidade com esta área do conhecimento e segundo pelo caráter social e tecnológico deste saber.

Esse trabalho é composto por seis tópicos com o objetivo de proporcionar uma melhor visão sobre a solução proposta. O mesmo divide-se em:

- Trabalhos Relacionados, onde será melhor abordado sobre Telerobótica e alguns trabalhos com mesmo objetivo do Projeto Controle;
- Python, que apontará algumas das vantagens no uso dessa linguagem no desenvolvimento de uma aplicação;
- Arduino, irá expor um pouco mais sobre essa placa de circuitos de plataforma livre;
- Estrutura e Arquitetura, mostrará como a integração do Python e Arduino podem ser transformados em um protótipo de Telerobótica;
- Considerações Finais virão com o objetivo de realizar uma análise geral da arquitetura, resultados alcançados com a estrutura obtida até agora, dificuldades encontradas no desenvolvimento e trabalhos que poderão ser desenvolvidos a partir da proposta apresentada

2. Trabalhos Relacionados

Além do espaço, a Telerobótica pode ser usada em distâncias menores, principalmente no que tange em salvamento de vidas. Os sistemas cirúrgicos Telerobóticos Zeus e da Vinci permitem a cirurgiões, a realização de cirurgias através de uma visão remota [3].

Existem vários programas que possibilitam o controle de um robô através de uma rede e no Brasil pode ser achado um deles. O RobWebCam (Figura 1) desenvolvido pelo Grupo de Automação e Controle Departamento de Engenharia Mecânica - GRACO, da Faculdade de Tecnologia da UNB, é um sistema composto por um manipulador com 2 graus de liberdade, que significa as capacidades de movimento em um plano/ambiente tridimensional, câmera de vídeo, conexão com a rede Internet, computadores e os drivers de comunicação entre os diversos componentes [4]. Seus desenvolvedores disponibilizam um site para teste desse sistema por 30 segundos, acessível em http://www.graco.unb.br/robwebcam.html.

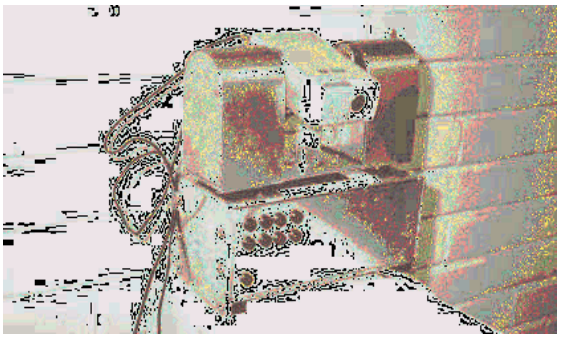


Figura 1 - RobWebCam

Fonte: http://www.graco.unb.br/robwebcam.html

Outro trabalho relacionado que pode ser citado nesta pesquisa é o *Curiosity* (Figura 2), veículo mecânico, construído para estar presente em ambientes hostis, como por exemplo a superfície de outro planeta. Apesar do *Curiosity* ter sido construído a partir de estudos diferentes dos realizados neste trabalho, é possível afirmar que o Protótipo de Telerobótica, denominado nesta pesquisa de Projeto Controle, pode alcançar objetivos semelhantes aos do veículo, na medida em que foi projetado para frequentar ambientes de difícil acesso humano.

O **MSL (Mars Science Laboratory)**, sonda especial lançada no dia 26 de novembro de 2011, pela Nasa. Dentro dessa sonda foi levado o ***Curiosity***, um veículo mecânico para a exploração do solo marciano. O equipamento espacial posou em Marte no dia 6 de agosto de 2012, mais precisamente na cratera Gale, após viajar no foguete Atlas V.[...] O curiosity utiliza câmeras de navegação batizadas de Navcams, equipamento instalado num mastro de metal com a possibilidade de captar imagem em 360º. O local de trabalho do novo robô fica nas redondezas da Cratera Gale ao sul do equador marciano, região do tamanho dos estados americanos de Connecticut e Rhode Island juntos. (REBOUÇAS, 2012)

Desenvolvido para realizar pesquisas em ambientes, em que a presença humana, nos dias de hoje ainda se torna impossível, o Curiosity faz parte dos avanços tecnológicos da sociedade contemporânea.

Figura 2 - Curiosity

Fonte: http://info.abril.com.br/noticias/ciencia/curiosity-pode-ter-feito-descoberta-historica-em-marte-21112012-42.shl

3. Python

I Python, em um propósito geral, é uma linguagem de programação de computador de código aberto. É otimizada para qualidade de software, a produtividade do desenvolvedor, a portabilidade do programa e integração de componentes. Python é usada por pelo menos centenas de milhares de desenvolvedores ao redor do mundo em áreas como script de Internet, programação de sistemas, interfaces de usuário, personalização de produtos, programação numérica, e muito mais. Considerada por estar entre as quatro ou cinco primeiras linguagens de programação mais utilizadas no mundo de hoje. Como uma linguagem popular, focada em diminuir o tempo de desenvolvimento, Python é usada em uma grande variedade de produtos e funções. Entre sua base de usuários atual estão o Google, YouTube, *Industrial Light & Magic*, o sistema de compartilhamento de arquivos BitTorrent, *Jet Propulsion Lab* da NASA, o jogo Eve Online, e o Serviço Nacional de Meteorologia dos Estado Unidos.

Python pode ser encontrado em uma ampla variedade de sistemas, contribuindo para seu crescimento rápido e contínuo no domínio da computação de hoje. Python é escrito em C, e por causa da portabilidade do C, Python está disponível em praticamente todo tipo de plataforma que tem um compilador ANSI C. Embora existam alguns módulos específicos de cada plataforma, geralmente qualquer aplicação escrita em Python em um sistema será executada com pouca ou nenhuma modificação na execução em outro sistema.

4. Arduino

A placa Arduino vem sendo utilizada com muito sucesso. A plataforma para o desenvolvimento dos programas de controle está disponível na Internet e existem diferentes versões do circuito no mercado nacional por preços acessíveis quando comparados às interfaces disponibilizadas no mercado por empresas como CIDEPE, Pasco Scientific e Phywe, por exemplo.

O Arduino, exibido na Figura 3, é uma ferramenta capaz de controlar, executar e interagir com o mundo físico, através de sensores e atuadores. O grande diferencial desta ferramenta é a sua utilidade e praticidade, pois são acessíveis e apresentam baixo

custo, bem como seu manuseio é, de certa maneira, de fácil acesso, tanto para experientes na área, quanto para amadores do ramo computacional.

Figura 3 - Arduino

Representando uma Plataforma física de computação livre, o Arduino é baseado numa placa de circuitos com um micro controlador, que pode ser programado na linguagem C de acordo com a função que executará, permitindo assim, a interação do software com o ambiente, através de sensores e atuadores.

Dentre as aplicabilidades do Arduino é possível citar o desenvolvimento de objetos interativos, estes que admitem:

Acoplagem de uma série de sensores ou chaves, que controlam uma variedade de luzes, motores ou outras saídas físicas. Projetos do Arduino podem ser independentes, ou podem se comunicar com software rodando em seu computador (como Flash, Processing, MaxMSP). Os circuitos podem ser montados à mão ou comprados pré-montados; o software de programação de código-livre pode ser baixado de graça [8].

Diante do exposto, observa-se que o Arduino veio a contribuir para o avanço tecnológico nos diversos sentidos, na medida em que propicia baixo custo, fácil acesso, além de ser uma plataforma que pode executar ações, de forma precisa e objetiva, unido a controladores de um ambiente, principalmente hostil, sendo muito útil em prototipagem de projetos.

Convém mencionar que o Arduino pode ser projetado e adaptado às necessidades de seu programador, por meio do incremento de placas chamadas *shields*, que são responsáveis pelo exercício de tarefas específicas como comunicação através de rede, seja ela com fio ou não, sensores de umidade, temperatura, rotação, aceleração, distância, manipulação de voltagens grandes o suficiente para danificá-lo usando relés, controladores de motores de passo ou de corrente contínua, dentre outros.

5. Estrutura e Arquitetura

O projeto desenvolvido neste trabalho consiste num sistema criado para permitir a interação de um usuário controlador com um ambiente qualquer remotamente, de maneira que o usuário deve estar em posse de um computador conectado com um cabo de ethernet a um roteador sem fio. Que por sua vez deve estar conectado a outro computador através da rede wireless, este, com uma câmera plugada para se extrair as imagens a serem enviadas. Assim ilustra a Figura 4.

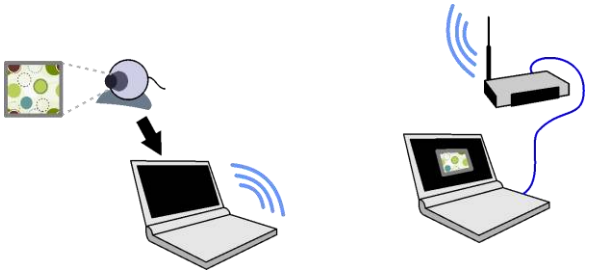


Figura 4 - Arquitetura física do Projeto Controle

No desenvolvimento do trabalho, de início foi necessário pesquisar sobre duas ações diferentes, que são básicas para o funcionamento do projeto com sucesso. As ações capturaram imagens da webcam e realizaram comunicação entre dois computadores através da rede, com o uso do Python. Estas ações foram escolhidas para este estudo, por curiosidade e pelo desafio de se aprofundar na linguagem de programação Python.

Ao longo do experimento, surgiu a ideia de integrar os estudos de captura de imagens ao de comunicação entre computadores, ou seja, desenvolveu-se toda a parte lógica da pesquisa que ao ser concluída, pode ser aplicada em diversas situações no uso de monitoramento com câmeras. É possível o uso da aplicação desenvolvida em sistemas de segurança com visualização em tempo real através da internet como sistema de trânsito ou sistemas internos de TV. Pode ser empregado, também, no desarmamento de bombas, análise de construções com risco de desabamento, cirurgias realizadas de maneira remota (Telecirurgia), dentre outros.

A aplicação desenvolvida necessita de uma configuração para ser executada. A mesma exige que seja informado um endereço de IP e uma porta de comunicação para ser estabelecida a conexão entre Cliente (Figura 5) e Servidor (Figura 6). Contudo, a versão Servidor disponibiliza, a quem o configura, informações que identificam se a webcam e o Arduino estão conectados e configurados corretamente.

O Projeto Controle Cliente possui dois campos onde o usuário deve digitar o IP que foi colocado no computador hospedeiro da versão Servidor e a porta de comunicação respectivamente. Após informar os dados anteriores, o botão OK deve ser clicado para realizar a tentativa de conexão. Se a tentativa de conexão obtiver sucesso, a janela inicial será desativada e será aberta uma nova janela (Figura 7), melhor explicada mais adiante, onde serão exibidos as imagens recebidas pelo Servidor e alguns botões para salvar imagem e outros para gravar vídeo e interromper essa gravação. Caso for necessário encerrar o programa, o botão Sair deverá ser acionado.

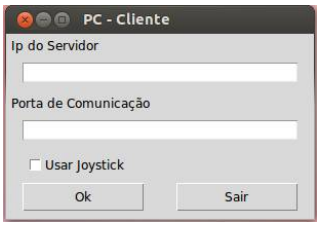


Figura 5 - Projeto Controle Cliente

A versão Servidor do programa também possui campos para que sejam informados o IP e a porta de comunicação. Ainda assim, possui um botão Detectar Dispositivo que, ao ser clicado, a webcam é configurada automaticamente e assim mostrado o caminho do dispositivo de vídeo no campo Dispositivo de Captura de Vídeo. O botão Detectar Arduino ativa uma função que configura o Arduino onde o símbolo dele se mantém monocromático enquanto o mesmo não estiver pronto para executar. Este símbolo fica da cor azul assim que o software julga que o Arduino está pronto para funcionar.

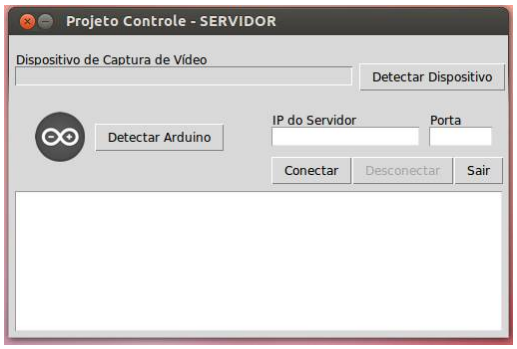


Figura 6 - Projeto Controle Servidor

A tela de execução do Projeto Controle Cliente (Figura 7) consiste em uma janela onde serão exibidas as imagens recebidas. A mesma possui, ainda, cinco botões onde suas funções são: salvar imagens visualizadas atualmente, iniciar a gravação de vídeo (círculo vermelho), parar a gravação de vídeo (quadrado cinza), escolher o local onde as imagens e os vídeos serão salvos (Configuração) e finalizar o programa (Sair).

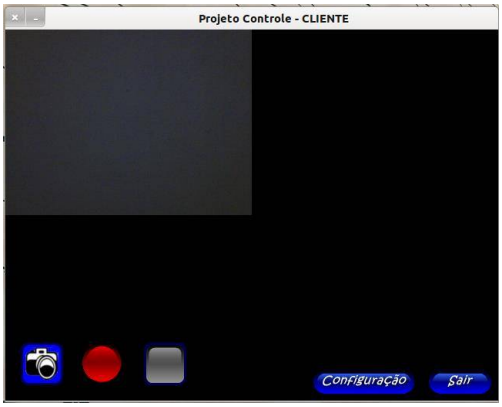


Figura 7 - Tela de execução do Projeto Controle Cliente

O software proposto, quando complementando com equipamentos robóticos, hardwares, placas de circuitos, motores, torna possível a construção de leques de possibilidades no ramo da Telerobótica. Dentre elas, foi escolhida a execução de Teleoperação de veículos automotores de quatro rodas, pela viabilidade de baixo custo, além de ser uma meta realizável a médio prazo. A Figura 8 ilustra a primeira versão do protótipo realizado.

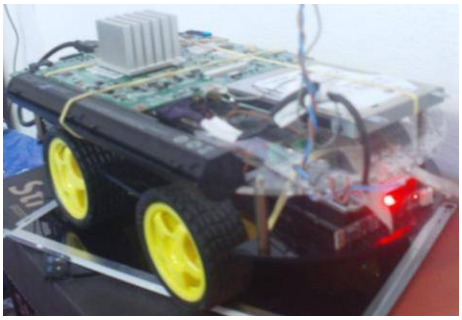


Figura 8 - Primeira versão do protótipo de Telerobótica: Projeto Controle

Percebeu-se que a partir da curiosidade de integrar as ações: captura de imagens com a comunicação de computadores, atrelando isso à Robótica, surgiu à resolução de inúmeros problemas, tais como: desarmamento de bombas, sem que o militar corra risco de vida; explorar planetas, a partir da atmosfera, ou seja, sendo desnecessário o contato direto do astronauta com o planeta explorado, evitando riscos de vida, Telecirurgia, monitoramento de domicílio por meio da internet, dentre outros.

O sistema Telerobótico proposto é composto basicamente por duas partes, o ambiente operador e o ambiente remoto, estes conectados através de uma rede de computadores qualquer, seja ela privada ou não, sendo necessária uma velocidade de operação considerável, para que haja a mínima perda de tempo possível, esse ponto sendo primordial ao desempenho da execução.

O ambiente operador é controlado por um usuário com um conhecimento intermediário em rede de computadores indispensável para a configuração do mesmo, responsabilizando-se por enviar caracteres que representam o estado atual da vontade do usuário (executar alguma ação ou não) oriundo de um dispositivo de entrada como: teclado, *joypad*, *joystick*, dentre outros. Após o envio desse comando o ambiente operador, põe-se em modo de escuta, preparado para receber dados que representarão uma imagem capturada da câmera acoplada ao robô.

O ambiente remoto deve conter uma câmera, além de apresentar possibilidade de locomoção (rodas, esteiras, etc.), onde são capturadas imagens a serem enviadas ao ambiente operador podendo ser pré-processadas para equilibrar o nível de carga de processamento das informações em ambos os ambientes, exigindo um processador e uma memória primária, representados neste trabalho por um Netbook disponibilizando recursos essenciais como câmera, conexão com rede wireless, entre outros. Ao ser enviada a imagem, o robô espera por receber uma resposta de seu operador, que representa tanto uma resposta de que o dado foi recebido, quanto o comando a ser executado, formando assim, uma comunicação half-duplex com o usuário.

Os ambientes, operador e remoto, devem manter-se conectados por uma velocidade de transmissão diretamente dependente do tamanho do quadro capturado, onde comprovado através de testes que a taxa de bits necessária para a transmissão de vídeo utilizando uma resolução de 320x240 pixels, mantém-se por volta de 1,5 Mbps, já se a resolução for estendida para 640x480, uma rede que proporcione 10 Mbps com facilidade será exigida para que o vídeo tenha desempenho satisfatório.

O realismo da imagem está relacionado com nossa maneira de perceber ambientes, que é baseado em gradientes de textura, projeções de objetos, reflexos de luz, sombras e assim por diante. Características destes dados implicam a transmissão de imagens de alta qualidade que requerem uma grande largura de banda. [...] a largura de banda mínima é de 5,2 Mbps. Corresponde às imagens comuns compactados. A largura de banda máxima é de 344 Mbps, o que corresponde a imagens estereoscópicas com 70 quadros por segundo e uma resolução de 640x480 pixels por imagem. (ARACIL *et al*, 2007, traduzido pelo autor).

Uma velocidade de aproximadamente 1,5 Mbps é necessária na transmissão de imagens com resolução de 320x240 pixels usando a aplicação que originou este trabalho. A estabilidade da velocidade da rede implica em um desempenho satisfatório, com uma taxa de frames por segundo (fps) próxima ao ideal, que quanto maior esta taxa, maior será a sensação de movimento que uma sequência de imagens pode propor e maior a quantidade de dados que serão enviados.

Diferente do que muitos imaginam a possibilidade de se desenvolver um robô, mesmo que simples, não está a uma distância muito grande. A robótica está envolvida principalmente com placas e equipamentos físicos, e não diferente de software, existe hardware livre. Juntando-se estes dois componentes, torna-se possível a concretização de idéias que integradas a dedicação e incentivo, atingem seus objetivos.

Neste trabalho, são usados o Arduino integrado a um *shield* que possui a tarefa de controlar motores elétricos de corrente contínua (Figura 9), estes usados no presente estudo para controlar movimentos direcionais (simulando um veículo automotor de quatro rodas), permitindo dessa forma ao Arduino um controle de voltagens, bem maiores que a voltagem disponibilizada pelo próprio Arduino, necessárias para a alimentação dos motores.

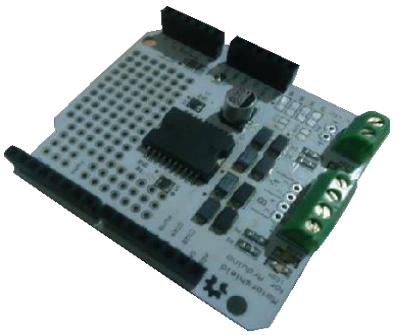


Figura 9 - *Shield* controlador de motores DC[1]

Em suma, utilizou-se o Arduino, com a finalidade de controlar os motores do experimento, no caso deste trabalho, atrelados a um *chassi* feito de acrílico de um carrinho que deve ter algumas peças de um Netbook necessárias para seu funcionamento como placa-mãe, HD e bateria, além do Arduino, o *shield* controlador dos motores, e uma fonte de energia para os motores independente da fonte do Netbook para serem capazes de executar o programa aqui proposto sendo possível, assim, sua execução onde essas peças estarão montadas em um chassi de acrílico como esquematizada na Figura 8.

[1] DC - Corrente Contínua

Sabe-se que um equipamento computacional precisa de uma parte lógica para que seja executada a tarefa desejada, nesse sentido, destaca-se que o microchip do Arduino, pode ser programado com a linguagem C, esta que realiza em seguida, o processamento dos dados.

Por ser programável na linguagem C, o Arduino permite a execução paralela com outras linguagens de programação que têm essa mesma base. Como o núcleo do Python é desenvolvido também em C, acaba que por proporcionar uma interação, entre a linguagem e o Arduino, extremamente facilitada, para isso, é indispensável o módulo Serial que deve ser instalado junto ao Python por não vir em sua biblioteca padrão.

6. Considerações Finais

O objetivo do presente trabalho foi apresentar uma arquitetura para Telerobótica. Foram exibidos alguns equipamentos que podem ser usados com esse propósito, além da sugestão de softwares nas versões Cliente e Servidor desenvolvidos em Python, que podem ser adaptados a qualquer tipo de serviço que envolva o monitoramento através de câmeras. Para isso, dedicou-se grande parte do tempo na realização de testes a fim de comprovar o funcionamento aplicável do projeto.

No desenvolvimento deste trabalho, o maior empecilho foi a falta de embasamento teórico. Não foi achado qualquer material que mostre uma integração de todas as áreas abordadas desenvolvidas na linguagem Python. A partir desse pressuposto, decidiu-se abordar as partes envolvidas na proposta separadamente. Assim foi mostrado que essas partes trabalhando em conjunto, poder-se-ia obter um sistema Telerobótico.

Devido à necessidade de equipamento de hardware na montagem do robô, também ocorreram dificuldades monetárias na aquisição das peças. Para se adquirir o equipamento com um preço menor, foi necessário comprar algumas das peças diretamente do exterior. Equipamentos de robótica comprados no país, podem chegar a valores de até cinco vezes maiores o valor do mesmo produto comprado de países como a China.

Após um ano de pesquisas e testes, a arquitetura proposta teve sua primeira versão funcional. Em geral, essa ideia teve bom resultados logo ao ser colocada em prática. A primeira versão do projeto obteve sucesso, mas ainda observou-se alguns detalhes que precisavam ser corrigidos.

O Netbook acoplado em cima do chassi de acrílico, mesmo sem sua carcaça, tornou-se muito pesado dificultando assim, o momento que o robô teria que realizar uma alteração de direção. Os pneus são feitos com uma borracha mole, aumentando o atrito entre os pneus e o chão, acarretando na dificuldade de se realizar uma curva. Deve ser levado em consideração que o chassi tem quatro pneus independentes, o que dificulta ainda mais a curva.

Ao se manter o Netbook desmontado, problemas como: falta de blindagem eletromagnética, perda de sinal da rede sem fio ou o risco de ter suas peças internas danificadas. Uma pessoa pode tocar o interior no Netbook e acarretar na queima de sua placa mãe por descarga de energia estática.

Além dos problemas mencionados anteriormente, observou-se uma instabilidade no momento em que o Python realiza comunicação serial com o Arduino. Pode ocorrer a situação em que o Python envie um comando para o Arduino sem que este esteja preparado para receber o dado. Acarretando no travamento total do sistema. Sendo necessário que o programa tenha seu processo morto através do gerenciador de processos. De consciência desses problemas apresentados, decidiu-se como trabalhos futuros, a substituição no Netbook por uma placa chamada Raspberry PI Model B, onde esta possui o tamanho de um cartão de crédito com um processador e memória que podem se aproximar a capacidade de processamento do Netbook.

Percebeu-se, ainda, que a velocidade de rede necessária para a transmissão das imagens é relativamente alta. Outra alteração futura a ser feita é a implementação de um algoritmo de compressão de dados para diminuir o tamanho do dado a ser transmitido. Ocasionando, assim, uma grande redução de velocidade diminuindo a carga no meio de transmissão. Outro ponto importante a ser modificado é o fato das rodas dianteiras dificultarem a mudança de direção do carro, podendo serem substituídas por uma pequena roda giratória que exerceria menos atrito com o chão. A proposta apresentada, portanto, atingiu seu objetivo. Ocorreram pequenos problemas já mencionados, que podem ser alterados posteriormente, dando um melhor aperfeiçoamento para a presente pesquisa.

References

J. M. S. Pinheiro, Quem é o Profissional de Telemática. Disponível em: <http://www.projetoderedes.com.br/artigos/artigo_quem_eh_o_profissional_de_telematica.php>. Acesso em: 30 mar. 2012.

N. Chopra, M. Ferre, A. Peer, *The Field of Telerobotics*. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=04658314>. Acesso em: 12 jan. 2012.

G. H. Ballantyne, *Robotic surgery, telerobotic surgery, telepresence & telementoring: Review of early clinical results*. Disponível em: <http://cmaps.cmappers.net/rid=1HZ2RWKZY-1Y1GHF0-G08/ResearchPaper6.pdf>. Acesso em: 15 fev. 2012.

A. J. Álvares; L. S. J. R. Junior Telerobótica: Metodologia para o desenvolvimento de sistemas robóticos teleoperados via Internet. Universidade de Brasília. Brasília. Disponível em: <ftp://ftp.graco.unb.br/pub/publicacoes_graco/aaacic.pdf>. Acesso em: 20 fev. 2012.

M. Lutz, *Programming Python*. 4.ed. Sebastopol: O'Reilly, 2011.

W. J. Chun, *Core Python Programming*. 2.ed. Upper Saddle River: Prentice Hall, 2006.

M. A. Cavalcante et al. Física com Arduino para iniciantes. Revista Brasileira de Ensino de Física. Disponível em: <http://www.scielo.br/pdf/rbef/v33n4/18.pdf>. Acesso em: 07 nov. 2011.

G. Araujo. O que é Arduino. Disponível em: <http://projeto39.wordpress.com/o-arduino/>. Acesso em: 16 out. 2011.