UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

FERRAMENTA DE SUPORTE A FORMAÇÃO DE REDES DE CONTATO NA BUSCA DE INFORMAÇÕES NA INTERNET

JANDIR BEPPLER

BLUMENAU
2006

2006/1-20

JANDIR BEPPLER

FERRAMENTA DE SUPORTE A FORMAÇÃO DE REDES DE CONTATO NA BUSCA DE INFORMAÇÕES NA INTERNET

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciências da Computação — Bacharelado.

Prof. Jomi Fred Hübner, Doutor - Orientador

BLUMENAU
2006

2006/1-20

FERRAMENTA DE SUPORTE A FORMAÇÃO DE REDES DE CONTATO NA BUSCA DE INFORMAÇÕES NA INTERNET

Por

JANDIR BEPPLER

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Jomi Fred Hübner – Orientador, FURB
Membro:	Prof. Paulo Fernando da Silva – FURB
Membro:	Prof. Roberto Heinzle – FURB

Blumenau, 10 de julho de 2006

Dedico este trabalho a todos os amigos, especialmente aqueles que me ajudaram diretamente na realização deste.

AGRADECIMENTOS

À Deus, pelo seu imenso amor e graça.

À minha família, pelo apoio incondicional.

Aos meus amigos, pelos empurrões e cobranças.

Ao Roosevelt, pelo incentivo na busca por novas tecnologias.

Ao meu orientador, Jomi Fred Hübner, por ter acreditado na conclusão deste trabalho.

“Jamais considere seus estudos como uma obrigação, mas como uma oportunidade invejável para aprender a conhecer a influência libertadora da beleza do reino do espírito, para seu próprio prazer pessoal e para proveito da comunidade à qual seu futuro trabalho pertencer.”

Albert Einstein

RESUMO

A sociedade dispõe de uma poderosa ferramenta para a comunicação e disseminação da informação, a Internet. Em constante evolução e expansão, ela cada vez mais disponibiliza novas fontes de informações sobre as mais diversas áreas. Porém esse contínuo crescimento está causando um aumento significativo no tempo necessário para que as informações sejam encontradas na rede. Fica claramente evidenciada a necessidade de haver ferramentas ou técnicas que viabilizem ao usuário a possibilidade de ter acesso mais rápido e otimizado às informações do seu interesse. Esse trabalho apresenta uma solução para esse problema, através do desenvolvimento de um assistente pessoal para os usuários de microcomputadores. O assistente terá como principal função, auxiliar os usuários no processo de pesquisa e obtenção dos dados contidos na Internet.

Palavras-chave: Rede de contatos. Agentes. Recuperação de informação.

ABSTRACT

The society has a powerful tool for the communication and dissemination of information, the Internet. In constant evolution and expansion, every day has new sources of information on the most several areas. However that continuous growth is causing a significant increase in the time necessary to find the information in the network. It’s necessary to develop tools or techniques that make possible the user the possibility to have faster access and optimized to the information of your interest. This work presents a solution for that problem, through a personal assistant development for microcomputers users. The principal function of the assistant is to aid the users in the research process and obtaining of the data contained in Internet.

Key-words: Contact network. Agents. Information recovery.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

XML – eXtensible Markup Language

UML – Unified Modeling Language

WWW – World Wide Web

WAN – Wide Area Network

W3C – World Wide Web Consortium

HTML – HyperText Markup Language

TCP – Transmission Control Protocol

HTTP – HyperText Transfer Protocol

EA – Enterprise Architect

SQL – Structured Query Language

SUMÁRIO

1 INTRODUÇÃO

"A Internet vem sendo qualificada como a ferramenta da nova sociedade globalizada e de aprendizagem por seu potencial de comunicação, disseminação da informação, e para o desenvolvimento econômico e social" (SILVA; THIRY; ABREU, 1999, p. 2). Paralelamente pode-se observar o aumento significativo da quantidade de informações disponíveis na Internet. Uma consulta por páginas de notícias esportivas, por exemplo, trará um número expressivo de possibilidades para que o interessado opte por uma ou mais que seja do seu interesse. Isso acarretará em um aumento significativo no tempo necessário para que a busca das informações desejadas seja concluída.

O fato da Internet dispor de uma grande quantidade de dados sobre as mais diversas áreas, sejam elas esportiva, entretenimento, financeira, etc, representa a viabilização ao acesso a essas informações por um público consideravelmente abrangente. No entanto, como foi citado anteriormente, o extraordinário crescimento da Internet acarretou um grande aumento da quantidade de dados disponíveis na grande rede, onerando o que inicialmente seria uma fonte rápida de acesso às informações. O problema consiste principalmente no expressivo número de fontes de informações disponíveis, ou seja, uma consulta por um determinado assunto efetuada por um usuário, raramente resultará na apresentação da resposta pretendida. Ao invés disso, um número considerável de páginas da Internet relacionadas ao tema da consulta será apresentada ao usuário. Em muitos casos, o conteúdo das páginas apresentadas não está relacionado com o significado do tema da pesquisa, portanto elas não serão úteis e mesmo assim são apresentadas.

Objetivando obter os dados ou as informações da Internet que realmente satisfaçam as necessidades dos usuários, no decorrer desse texto será apresentada uma implementação de uma ferramenta de suporte a formação de redes de contato para recuperação de informações

da grande rede. Como o próprio nome sugere, a função da ferramenta será a de ser uma facilitadora, ou seja, ela será responsável por otimizar o processo de pesquisa na Internet. A ferramenta observará as pesquisas realizadas pelo usuário e armazenará o resultado das buscas em uma base de dados do usuário em questão. Para as buscas futuras ela levará em consideração o conteúdo dessa base de dados e também da base de dados dos demais usuários que forem membros da rede de contatos previamente formada, e só então ela fará a pesquisa na Internet. Os usuários possuem perfis, ou seja, um profissional de informática faz pesquisas relacionadas com esse tema e ele se relaciona com pessoas da mesma área, e assim por diante. É baseado nessa linha de raciocínio que a ferramenta será desenvolvida.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é desenvolver um sistema multi-agentes formado por agentes de informação ou de Internet, voltados para recuperação de informações contidas na grande rede. O sistema será capaz de encontrar informações relevantes para o usuário, baseado nos resultados das pesquisas que este realizou previamente e também baseado nas pesquisas realizadas por outros usuários pertencentes ao sistema. Em outras palavras, cada usuário pertencente ao sistema terá seu próprio agente, sendo assim, agentes de diferentes usuários trocarão informações a fim de atender às solicitações que lhes foram propostas.

1.2 ESTRUTURA DO TRABALHO

Esse trabalho está estruturado como descrito a seguir:

a) no capítulo 2, serão apresentadas todas as áreas relacionadas com a fundamentação teórica, necessárias para o desenvolvimento desse trabalho, sendo elas: Agentes, .Net Remoting e XML;

b) no capítulo 3 é apresentado todo o processo de desenvolvimento do software, incluindo os requisitos, algoritmos, técnicas, diagramas, trechos de códigos, etc, utilizados durante esta etapa;
c) no capítulo 4 serão apresentadas conclusões, análise dos resultados obtidos e sugestões para extensão do presente trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Esse capítulo está organizado em quatro seções. A primeira apresenta definições e as principais características dos agentes. A segunda seção apresenta a tecnologia .Net Remoting. A terceira seção apresenta algumas características e funcionalidades da tecnologia XML utilizadas no desenvolvimento deste trabalho. Por fim a quarta seção apresenta os trabalhos correlatos.

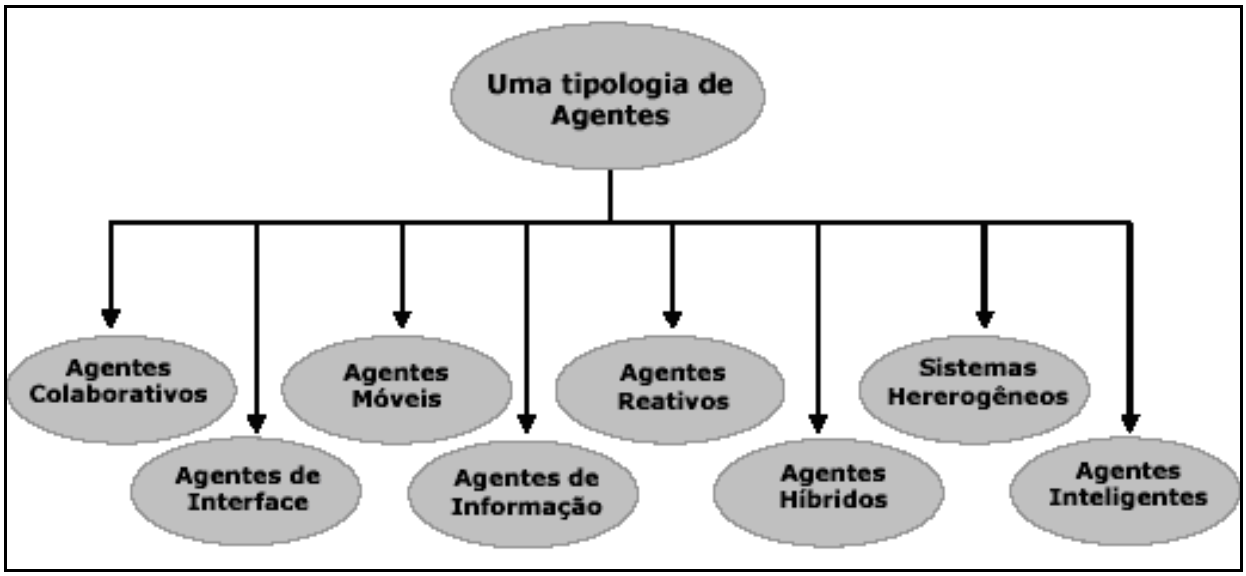
2.1 AGENTES

Ainda não é possível encontrar um termo para definir com exatidão o propósito dos agentes, devido a sua aplicabilidade em diversas áreas do conhecimento e a imaturidade da área. Pattie Maes, uma das principais pesquisadoras dessa área, define agentes como sendo sistemas de computadores desenvolvidos para atuarem em ambientes dinâmicos, capazes de identificar mudanças nesses ambientes e a partir dessas mudanças realizarem as tarefas para as quais foram desenvolvidos (MAES, 1995 apud VAHL, 2003, p. 58). Assim como Pattie Maes, outros pesquisadores também apresentam suas definições e entendimentos sobre os agentes, contudo essas definições geralmente tendem a representar uma visão teórica focada na área de atuação do pesquisador em questão.

Saindo um pouco do âmbito teórico dessas definições, observa-se uma visão igualmente importante como a citada anteriormente, porém por tratar-se de um entendimento apresentado por uma empresa desenvolvedora de sistemas, talvez essa definição seja melhor compreendida: agentes são entidades que realizam um conjunto de tarefas ou operações para as quais foram projetadas, munidas com algum grau de independência ou de autonomia, empregando, para isso, algum entendimento ou representação dos objetivos ou desejos do

usuário (IBM CORPORATION, 2003, P. 10).

Como foi citado anteriormente, os agentes são aplicáveis nas mais diversas áreas. Portanto, para o seu desenvolvimento deverá ser observado qual delas mais se aproxima da aplicação em questão. Nwana (1996 apud VAHL, 2003, p. 65) apresenta um modelo ou uma tipologia de classificação de agentes.



Fonte: Nwana (1996 apud VAHL, 2003, p. 65).

Figura 1 – Classificação dos agentes

De acordo com a tipologia apresentada na Figura 1, as características de cada uma das classes de agentes são as seguintes:

a) agentes colaborativos: cooperam com outros agentes para realizar as tarefas que lhes foram incumbidas;
b) agentes de interface: esses agentes auxiliam o usuário, observando suas ações e imitando-as, ou aprendem por meio de instruções explícitas do usuário;
c) agentes móveis: são capazes de percorrer uma WAN como a Internet, em busca de informações para o usuário;
d) agentes de informação/Internet: realizam a coleta de informações de várias fontes distribuídas presentes na WWW;
e) agentes reativos: funcionam através de um estímulo, devolvendo uma resposta (modelo de ação/reação);
f) agentes híbridos: são constituídos de duas ou mais filosofias de agentes;

g) sistemas heterogêneos: são sistemas constituídos de dois ou mais agentes com diferentes arquiteturas;
h) agentes inteligentes: seriam constituídos por todas as características citadas anteriormente, incluindo autonomia, aprendizagem e cooperação, mas o autor ressalta que, na realidade representam apenas inspiração dos pesquisadores.

A classificação dos agentes acima apresentada permite identificar as características, habilidades e capacidades que são inerentes a cada um deles. Na seção seguinte será apresentada em detalhes a classe dos Agentes de informação ou de Internet, que serão empregados no desenvolvimento desse trabalho.

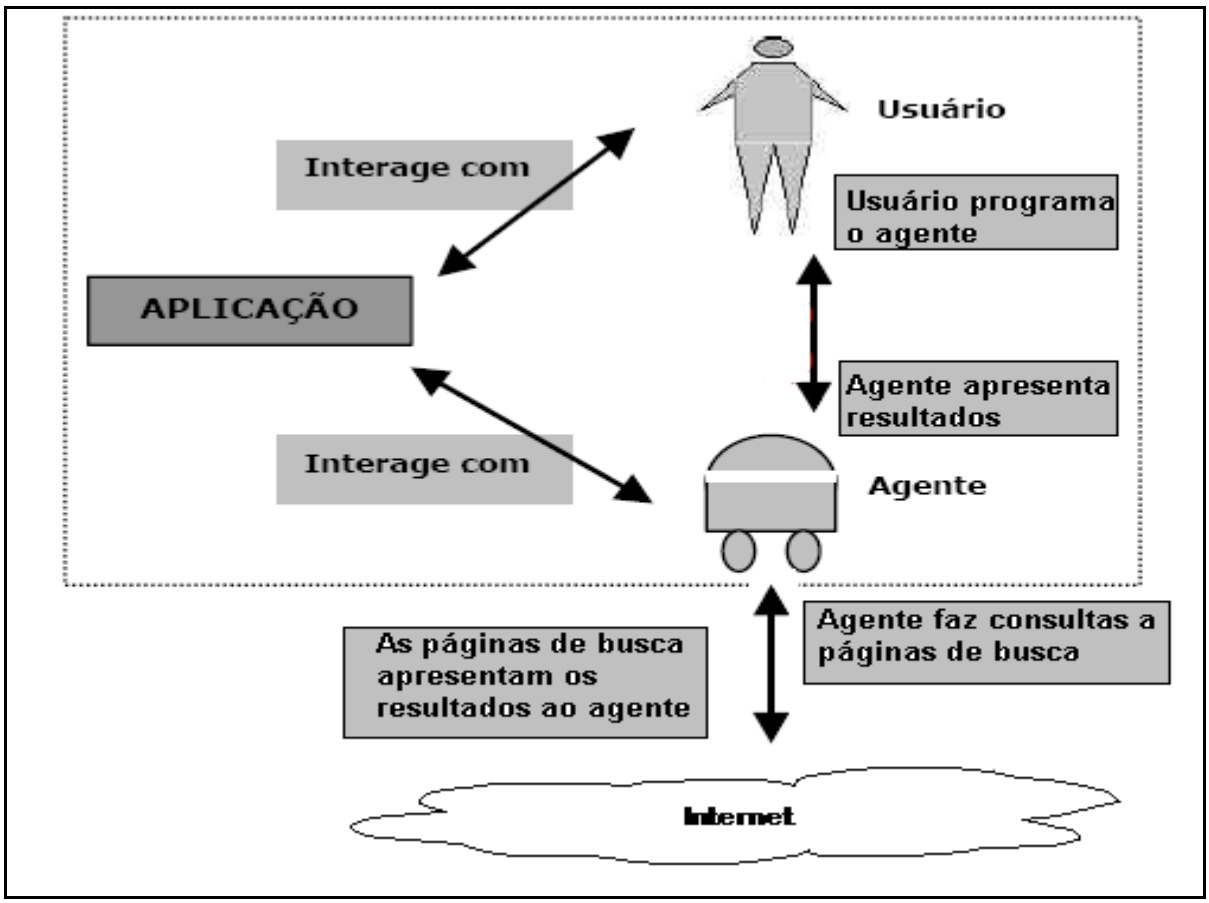
2.1.1 Agentes de informação ou de Internet

"[...] a melhor metáfora usada para descrever agentes é a de um assistente pessoal que colabora com o usuário no mesmo ambiente de trabalho" (MAES, 1995 apud VAHL, 2003, p. 58). De acordo com essa definição apresentada por Pattie Maes, os agentes pertencentes a essa classe se caracterizam por prover acesso eficiente a fontes heterogêneas de informação. Entre elas pode-se citar a Internet, *Newsgroups* ou até mesmo um banco de dados disponível em uma rede.

De acordo com Nwana (1996 apud MAIA, 2004, p. 59), a principal característica inerente aos agentes de informação é a que eles tornam transparente aos usuários a complexidade e heterogeneidade do acesso à informações contidas em sistemas distribuídos e fornecem meios eficientes para que os resultados encontrados sejam apresentados.

Uma aplicação que engloba os conceitos apresentados é o Copernic (COPERNIC TECHNOLOGIES INC, 2006). O Copernic é composto por um agente responsável por executar buscas em várias páginas de pesquisa na Internet como o Google, Yahoo e o Altavista e exibir os resultados ao usuário. Essa aplicação apresenta com clareza as características da classe de agentes abordada por esse trabalho, ou seja, ao invés do usuário

acessar diretamente as páginas de busca, esse trabalho é feito pelo agente tornando esse processo mais eficaz e produtivo visto que o agente acessa as principais fontes de informações presentes na Internet. A Figura 2 apresenta o funcionamento de um agente de informação ou de Internet.



Fonte: adaptado de Vahl (2003, p. 67).

Figura 2 – Agentes de informação ou de Internet

A descrição das ações da Figura 2 apresentam-se da seguinte forma: o usuário informa ao sistema informações referentes as pesquisas a serem realizadas e também configura o agente para monitorar determinadas páginas da Internet. De posse dessas informações o agente faz buscas na Internet, apresenta os resultados ao usuário e dependendo da configuração feita pelo usuário ele pode armazenar os resultados das buscas realizadas para serem visualizadas posteriormente.

"O objetivo não é simplesmente encontrar informações que satisfaçam um conjunto de palavras-chave, mas espera-se que este tipo de agente possa reconhecer padrões de

informação e encontrar aquelas mais relevantes” (COSTA, 1999). Além de executar as tarefas solicitadas pelo usuário, que nesse caso consistem em buscas na Internet, o agente também deve ter a capacidade de identificar padrões das pesquisas realizadas, para que em futuras pesquisas ele possa otimizá-las tornando esse processo mais rápido.

Os agentes de informação ou de Internet se destacam no seu propósito por apresentarem características variadas, ou seja, eles podem ser móveis e percorrerem redes em busca de informações que lhes foram solicitadas e posteriormente retornarem com os resultados obtidos, ou podem ser estáticos sendo que nesse caso, todo o processo de pesquisa das informações será realizado a partir do computador do usuário (SILVA, 2002, p. 27). Embora os grandes portais de buscas da Internet como o Google, Yahoo e o Altavista sejam capazes de fornecer uma quantidade considerável de informações durante as buscas, eles não possuem qualquer mecanismo de filtragem para essas informações e é justamente essa deficiência que os agentes de informação ou de internet buscam suprir.

2.2 .NET REMOTING

A tecnologia .Net Remoting é um *middleware* para comunicação entre sistemas distribuídos, caracterizada principalmente por prover um alto grau de transparência na comunicação entre objetos. E permite que o programador se abstraia de primitivas básicas de comunicação, trazendo grandes vantagens em termos de desenvolvimento e manutenção das aplicações (MICROSOFT CORPORATION, 2006).

A estrutura da comunicação dos sistemas distribuídos se assemelha em grande parte com a arquitetura convencional cliente-servidor. No entanto elas diferem em um ponto crucial, que é na maneira com que as requisições são efetuadas e atendidas pelas partes envolvidas. Na arquitetura convencional cliente-servidor, um servidor de arquivos, por

exemplo, o cliente solicita um determinado arquivo ao servidor, esse por sua vez fornece o arquivo para ser processado no cliente. Já os sistemas distribuídos se caracterizam por fazerem chamadas para métodos remotos, ou seja, o cliente efetua uma requisição a um objeto remoto, esse efetua o processamento necessário, e devolve para o cliente o resultado da requisição (LOTH, 2005. p. 5).

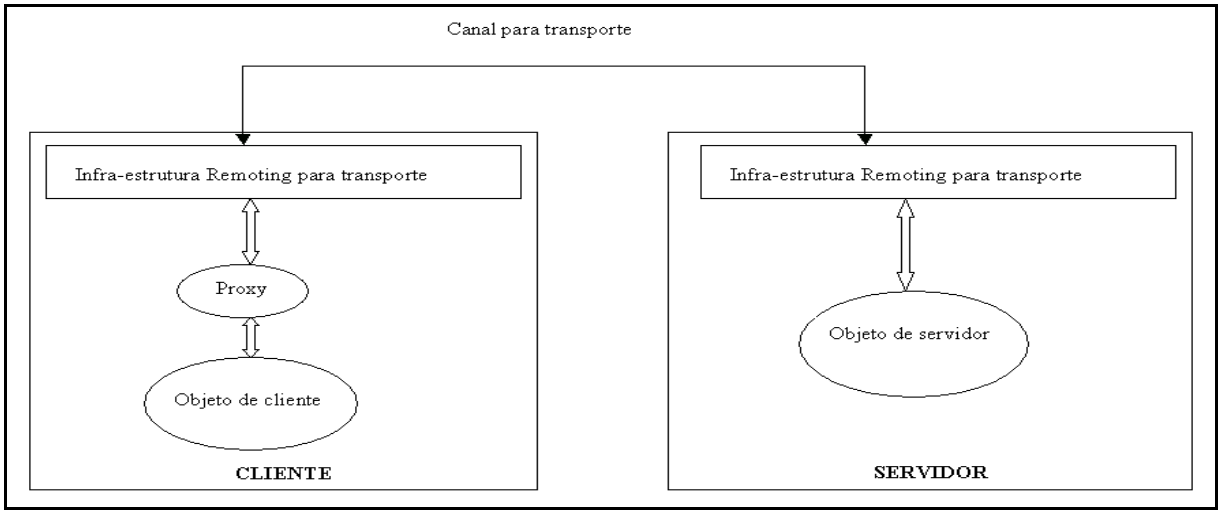
2.2.1 Arquitetura do .Net Remoting

Como foi citado anteriormente, a comunicação é o pilar de sustentação de qualquer sistema distribuído. O .NET Remoting permite separar o objeto remoto da aplicação e também, abstrair o mecanismo de comunicação utilizado. De acordo com Robinson et al. (2001, p. 981), a arquitetura .NET Remoting se caracteriza pela utilização dos seguintes elementos:

a) um objeto remoto: os objetos remotos possuem a capacidade de serem invocados por outro processo. Eles são classificados em dois tipos, o *Marshall-by-value*, caracterizado por passar uma cópia do objeto ao solicitante, permitindo a invocação de seus métodos diretamente como qualquer outro objeto local. E o segundo tipo é o *Marshall-by-reference*, que passa uma referência do objeto ao solicitante, instanciando-o remotamente;
b) um servidor: para hospedar o objeto e aguardar as requisições por ele;
c) um cliente: para fazer requisições para o objeto remoto.

A flexibilidade da arquitetura pode ser verificada nos seus elementos apresentados anteriormente. O programador pode implementar a comunicação entre os elementos citados, através da utilização de dois canais de transporte, sendo eles: o *HTTPChanel* caracterizado pela utilização do protocolo HTTP serializado no formato XML e também por permitir que as aplicações se comuniquem sem que haja a necessidade de abertura de portas em *firewalls*. E o

TCPChanel caracterizado por transportar os dados utilizando o protocolo TCP serializado no formato binário (LOTH, 2005. p. 11).



Fonte: adaptado de Microsoft Corporation (2006).

Figura 3 – Arquitetura do .NET Remoting

A Figura 3 apresenta a forma geral de como é realizada uma chamada remota. Esse exemplo poderia ser usado para descrever uma aplicação sendo executada em um determinado computador, e que está usando uma funcionalidade que é provida por um objeto que está em execução em outro computador.

Robinson et al. (2001, p. 978) descreve os eventos apresentados pela Figura 3 da maneira como segue. A classe cliente instancia a classe servidor e a infra-estrutura *Remoting* para transporte se encarrega dos demais passos. A classe cliente cria também um objeto *Proxy*, responsável pela representação da classe cliente e também é encarregado de passar as chamadas dos métodos para o servidor.

Quando o processo é iniciado, a classe cliente solicita um método, a infra-estrutura *Remoting* para transporte recebe essa solicitação e a envia através do canal para transporte para ser processada na classe servidor. Quando a requisição chega ao servidor, a infra-estrutura *Remoting* para transporte a interpreta e cria as instâncias para que método seja executado. Após o término do processamento pelo servidor, tem início o processo de resposta para o cliente. A infra-estrutura *Remoting* para transporte empacota a resposta e a enviada

para o cliente através do canal para transporte.

2.2.2 Um exemplo prático da arquitetura

O exemplo a seguir, foi desenvolvido na plataforma .Net utilizando a linguagem C#. O exemplo é simples, porém engloba as principais funcionalidades da tecnologia. O desenvolvimento do exemplo se inicia com a implementação da interface do projeto, na seqüência tem-se o desenvolvimento do servidor e por fim apresenta-se o cliente.

"Uma interface é como se fosse um controle remoto. Você consegue interagir com um objeto remoto conhecendo o que ele oferece, tendo a interface que descreve cada função, porém, sem a mínima idéia de como ele implementa essa funcionalidade internamente" (PAULI, 2004). Segundo essa definição de Pauli, a interface será usada tanto pelo cliente como pelo servidor. O servidor a implementará e o cliente por sua vez, utilizará os serviços providos por ela.

```
using System;

namespace AtivadorDeProcessos
{
    /// <summary>
    /// descrição da interface AtivadorDeProcessos.
    /// </summary>
    public interface AtivadorDeProcessos
    {
        bool Run(string programa, string argumento);
    }
}
```

Fonte: adaptado de Robinson et al. (2001, p. 982).

Figura 4 – Interface

Observa-se que para o desenvolvimento da interface, não foi definido nenhum código para execução da mesma, ou seja, ela contém apenas descrição de classes. Dessa forma, ela permite que os objetos remotos utilizem todos os métodos definidos (LOTH, 2005. p. 9).

A seguir apresenta-se a implementação do servidor. Ele é derivado da interface, ou seja, ela a implementa e fica em execução aguardando as requisições dos clientes.

```
using System;
using System.Diagnostics;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Tcp;
using AtivadorDeProcessos;

namespace Servidor
{
    public class AtivadorDeProcessosServidor : MarshalByRefObject, AtivadorDeProcessos.AtivadorDeProcessos
    {
        public AtivadorDeProcessosServidor()
        {
            // TODO: Add constructor logic here
        }
        public bool Run(string programa, string argumento)
        {
            Process.Start(programa, argumento);

            return false;
        }
        public static void Main()
        {
            TcpServerChannel channel = new TcpServerChannel(9000);
            ChannelServices.RegisterChannel(channel);
            WellKnownServiceTypeEntry remObj = new WellKnownServiceTypeEntry
            (
                typeof(AtivadorDeProcessosServidor),
                "Servidor",
                WellKnownObjectMode.SingleCall
            );
            RemotingConfiguration.RegisterWellKnownServiceType(remObj);
            Console.WriteLine("Pressiome [ENTER] para sair.");
            Console.ReadLine();
        }
    }
}
```

Fonte: adaptado de Robinson et al. (2001, p. 983).

Figura 5 – Servidor

Para o desenvolvimento do servidor foram utilizadas algumas classes responsáveis pela comunicação. Robinson et al. (2001, p. 982) apresentam uma definição dessas classes como pode ser observado a seguir.

a) *TcpServerChannel*: classe responsável por criar um canal de comunicação TCP;
b) *ChannelServices.RegisterChannel*: registra os objetos remotos para permitir que seus métodos sejam acessados remotamente, ou seja, o servidor registra a localização dos objetos;
c) *RemotingConfiguration.RegisterWellKnownServiceType*: responsável por registrar junto ao servidor, um determinado tipo como sendo um objeto remoto;
d) *WellKnownObjectMode.SingleCall*: quando um objeto é ativado no servidor, é atribuído a ele um tempo de vida útil. Os objetos do tipo *SingleCall*, são criados a cada chamada estabelecida pelo cliente. Os objetos podem ser também do tipo *Singleton*, sendo que nesse caso eles são criados somente uma vez e todos os clientes se comunicam com o mesmo objeto.

Por fim apresenta-se a implementação do cliente, responsável por realizar as requisições remotas.

```
using System;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Tcp;
using AtivadorDeProcessos;

namespace Cliente
{
    class Cliente
    {

        [STAThread]
        static void Main(string[] args)
        {


            TcpClientChannel channel = new TcpClientChannel();

            ChannelServices.RegisterChannel(channel);

            AtivadorDeProcessos.AtivadorDeProcessos processo =
                (AtivadorDeProcessos.AtivadorDeProcessos)Activator.GetObject
            (
                typeof(AtivadorDeProcessos.AtivadorDeProcessos),
                "tcp://localhost:9000/AtivadorDeProcessosServidor"
            );

            processo.Run("IExplore.exe", @"http://www.furb.br");
        }
    }
}
```

Fonte: adaptado de Robinson et al. (2001, p. 983).

Figura 6 – Cliente

A arquitetura .NET Remoting através da utilização de *proxy*, permite que os clientes recebam referências remotas de um objeto servidor, e executem chamadas aos seus métodos, dando a impressão que eles estão sendo executados localmente (LOTH, 2005. p. 5).

2.3 XML

XML pode ser definido como uma linguagem de marcação para descrever dados estruturados. Desenvolvida em 1996 pelo W3C, essa linguagem permite ao desenvolvedor especificar ou armazenar diversos tipos de informações ou dados, entre eles podes-se citar a documentação de softwares, conteúdo de páginas da Internet, carga e descarga de banco de dados e os mais diversos documentos textuais (DEITEL; DEITEL et al., 2003, p. 724). A

linguagem XML herdou das linguagens de marcação a capacidade de criar documentos que possuem uma estrutura definida em forma de árvore, ou seja, trechos ou partes do documento são delimitados por marcadores também conhecidos por tags, esses marcadores permitem distinguir claramente onde iniciam e onde finalizam as partes que constituem o documento. Deixando assim o desenvolvedor livre para definir a estrutura do documento de acordo com a sua finalidade.

A Figura 7 apresenta um exemplo de documento XML.

```
- <usuarios>
  - <usuario nome="Jandir Beppler">
    - <argumentos>
      - <argumento args="Windows 2003">
        - <links>
            <link url="http://www.microsoft.com/Windowsserver2003/" />
            <link url="http://www.microsoft.com/windows/" />
            <link url="http://www.microsoft.com/Windowsserversystem/" />
          </links>
        </argumento>
      </argumentos>
    </usuario>
  </usuarios>
```

Figura 7 – Exemplo de documento XML

Segundo Dias (2004), os elementos de um documento XML são responsáveis pela descrição dos dados. Assim como a clara delimitação do início e fim do trecho designado a ele, tornando o documento legível e facilitando a manipulação e interpretação dos dados. No exemplo apresentado pela Figura 7, o elemento <usuarios></usuarios> delimita a totalidade do conteúdo do documento. Já os demais elementos aninhados, são responsáveis pela delimitação da sua área em específico, de acordo com a definição que lhe foi proposta.

“A XML existe porque a HTML teve sucesso. Portanto, a XML incorpora muitos recursos bem-sucedidos da HTML. Ela também existe porque a HTML não conseguiu acompanhar as novas demandas” (MARCHAL, 2000, p. 9). Embora seja uma linguagem amplamente difundida e com muitos recursos, a HTML não é extensível, consequentemente ela não está acompanhando a constante evolução da Internet, esse fator enfatiza a necessidade

de haver uma nova solução para a manipulação de dados estruturados, que seja capaz de manter a simplicidade no desenvolvimento, aliada a característica de ser extensível permitindo se adaptar a possíveis mudanças na estrutura dos documentos.

2.3.1 XML para armazenamento de dados

Originalmente a XML foi criada para suprir as necessidades de estruturação de documentos voltados para a Internet. No entanto, características de flexibilidade, independência de plataforma e por ser extensível, permitem a XML ser utilizada como ferramenta para manipulação de informações de banco de dados ou mesmo para o armazenamento dos dados (DIAS, 2004, p.16). Analisando essa definição, conclui-se que entre as vantagens de se utilizar XML para o armazenamento de dados, apresenta-se a facilidade para execução do sistema, ou seja, não há a necessidade de fazer qualquer instalação ou configuração para que as informações sejam salvas com a integridade necessária. Todo esse controle é feito pela aplicação, tornando o processo transparente para os usuários finais.

Os arquivos XML mantêm as informações altamente organizadas e delimitadas em seções, contudo é necessário que haja mecanismos para obter as informações contidas nesses arquivos. Em banco de dados tradicionais os sistemas usam expressões SQL para criar tabelas, extrair, inserir ou consultarem dados. A linguagem XML dispõe de um mecanismo para realização de pesquisas em seus arquivos denominado XPath, ele permite a realização do acesso aos nós dentro de um documento, utilizando expressões que contém o caminho dentro do arquivo onde se encontram os dados requeridos (DIAS, 2004, p. 25).

2.4 TRABALHOS CORRELATOS

Várias ferramentas que auxiliam os usuários de computadores estão disponíveis no mercado, contudo algumas se destacam pelas suas funcionalidades ou por serem fáceis de serem utilizadas. Recentemente chegou ao mercado o GoogleDesktop (GOOGLE INC, 2005). Através de sua interface, o usuário pode obter informações sobre o mercado financeiro, informações sobre a meteorologia e também visualizar as principais notícias em tempo real. Todas essas funcionalidades podem ser configuradas pelo usuário, bem como a maneira com que essas informações serão apresentadas.

Concorrentemente à ferramenta citada anteriormente, está disponível no mercado o assistente MSN Search (MICROSOFT CORPORATION, 2005). Através desse assistente, o usuário poderá efetuar buscas em seu computador por arquivos ou trecho do conteúdo de um arquivo e buscas por imagens e nesse caso o próprio assistente exibe o conteúdo. Também é possível efetuar buscas por páginas da Internet, nesse caso o usuário informa uma palavra e o assistente encontra as páginas relacionadas. Mas o grande destaque dessa ferramenta é que ela permite que o usuário cadastre páginas da Internet que disponibilizam notícias no formato RSS, ou seja, através de uma interface gráfica, é possível informar quais páginas que se deseja monitorar e o assistente encarrega-se de coletar as informações e posteriormente apresentá-las.

Aplicando os principais conceitos e características sobre agentes e principalmente sobre os agentes de informação ou de Internet encontra-se disponível em versão gratuita o agente Copernic (COPERNIC TECHNOLOGIES INC, 2006). O Copernic faz buscas simultâneas nos principais diretórios de informações da WWW, como Google, Yahoo, Altavista e Lycos. Os resultados são filtrados, evitando assim links repetidos e depois organizados por relevância, e apresentados na mesma tela. As funcionalidades deste programa

vão além. É possível realizar buscas por algo mais específico, como um Livro ou um software, basta realizar as configurações necessárias através da interface. Outro detalhe importante é que o Copernic permite o salvamento das buscas realizadas, otimizando as futuras buscas que forem efetuadas.

Uma ferramenta que também se destaca é o Aglets em um Sistema de Busca de Informações (ASBI), descrita em Lobato e Damasceno (2003). Essa ferramenta consiste em um sistema de pesquisa de preços de livros em uma rede local. O usuário pode informar, por exemplo, um título de um livro para o sistema. Esse por sua vez efetuará uma pesquisa de preços e posteriormente apresentará os resultados obtidos.

3 DESENVOLVIMENTO DO SISTEMA

Esse capítulo está dividido em cinco seções. A primeira delas apresenta os requisitos funcionais e não funcionais do sistema a ser desenvolvido. A segunda seção apresenta uma visão geral do funcionamento do sistema. A terceira seção apresenta a especificação do sistema desenvolvido através de diagramas que o representam logicamente. A quarta seção, aborda alguns aspectos da implementação do sistema. A quinta seção apresenta os resultados e discussões do trabalho.

3.1 REQUISITOS DO PROBLEMA A SER TRABALHADO

Os requisitos funcionais do sistema são: o sistema deve ser capaz de efetuar buscas na WWW baseado no tema para pesquisa informado pelo usuário; apresentar os resultados das buscas ao usuário; trocar informações com outros agentes do sistema; armazenar os resultados das buscas efetuadas em uma base de dados; realizar buscas na base de dados local; permitir a criação de grupos de contatos; permitir o cadastramento de usuários.

Os requisitos não funcionais do sistema são: utilização da plataforma de desenvolvimento .Net utilizando a linguagem C#; armazenamento dos dados em arquivos XML; para o funcionamento do sistema, os computadores deverão estar conectados à Internet ou em uma rede local.

3.2 VISÃO GERAL

A Figura 8 apresenta o funcionamento geral do sistema quando o usuário efetua uma pesquisa por um determinado assunto.

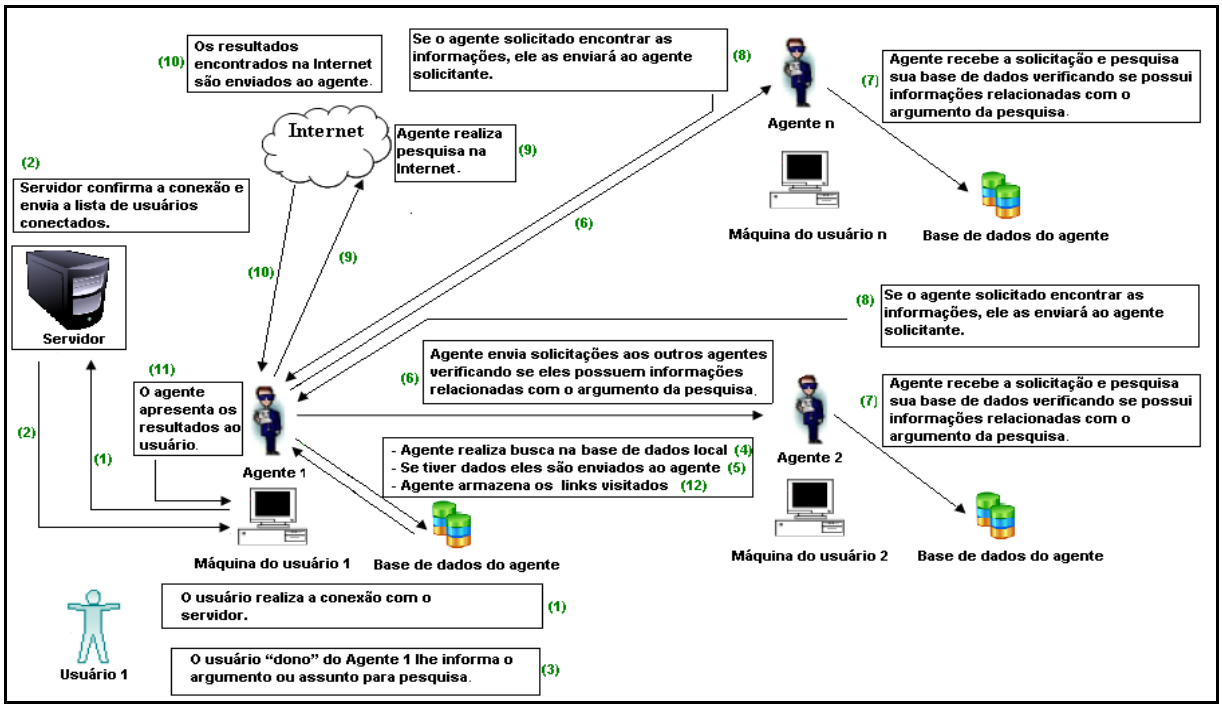


Figura 8 – Funcionamento geral do sistema durante uma busca

A descrição da seqüência das ações acima apresentadas dar-se-á da seguinte forma:

a) ação (1): o usuário informa seu nome e a senha para se conectar ao servidor;
b) ação (2): o servidor faz a autenticação do usuário e o envia a lista dos demais usuários conectados;
c) ação (3): o usuário informa um argumento ou assunto para que seu agente realize as pesquisas;
d) ação (4): o agente realiza primeiramente uma busca na sua base de dados;
e) ação (5): as informações encontradas na base de dados local são enviadas ao agente;
f) ação (6): o agente envia o argumento da pesquisa aos demais agentes conectados para que estes verifiquem se possuem informações relacionadas;
g) ação (7): os agentes recebem a solicitação e realizam uma busca na sua base de dados;
h) ação (8): as informações encontradas na base de dados local são enviadas ao agente solicitante;
i) ação (9): o agente realiza buscas na Internet;
j) ação (10): as informações encontradas na Internet são envidas ao agente;
k) ação (11): o agente apresenta os resultados das buscas ao usuário;
l) ação (12): os endereços das páginas visitadas pelo usuário são armazenados.

3.3 ESPECIFICAÇÃO

A especificação do sistema apresenta-se através dos diagramas da UML, utilizando para tal os diagramas de casos de uso, diagramas de classe e diagrama de atividades. Também apresenta-se a modelagem para o armazenamento do dados em arquivos XML.

3.3.1 Diagramas UML

Os diagramas da UML serão gerados com a utilização da ferramenta EA. Eles apresentam-se cada um em uma subseção como a seguir.

3.3.1.1 Diagramas de casos de uso

“Um caso de uso é uma descrição de um conjunto de seqüências de ações, inclusive variantes, que um sistema executa para produzir um resultado de valor observável por um ator” (BOOCH; RUMBAUGH; JACOBSON, 2000, p. 220). Sendo assim, essa modelagem é usada para descrever o que o sistema deve fazer, ou seja, os diagramas de casos de uso descrevem os requisitos funcionais do sistema.

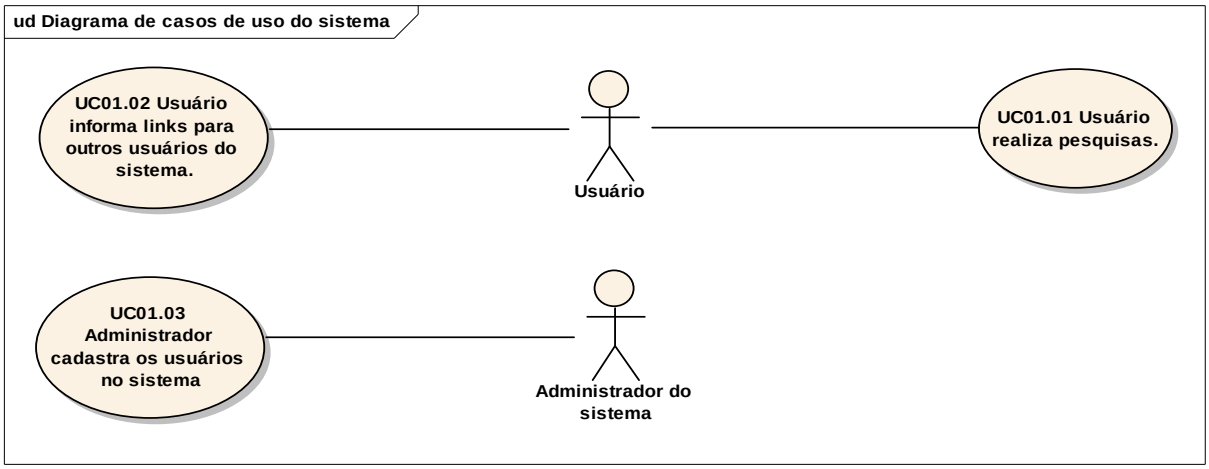


Figura 9 – Diagrama de casos de uso do sistema

O diagrama de casos de uso acima apresentado pode ser descrito da seguinte maneira: o usuário informa um argumento para pesquisa ou indica uma página da Internet para outro usuário; o agente recebe a solicitação do usuário, se for a indicação de uma página da Internet ele a envia para o destinatário. Se for um argumento ele iniciará o processo de pesquisa, partindo da base local em seguida efetua consultas aos outros agentes e por fim consultará a Internet; após o processo de pesquisa o agente apresentará os resultados obtidos ao usuário e as páginas que cujo conteúdo interessar ao usuário serão armazenadas em uma base de dados.

3.3.1.2 Diagrama de classes

"Um diagrama de classe é um diagrama que mostra um conjunto de classes, interfaces e colaborações e seus relacionamentos" (BOOCH; RUMBAUGH; JACOBSON, 2000, p. 105). Dessa forma, os diagramas de classes apresentam as classes que compõem um sistema bem como os relacionamentos entre elas.

A Figura 10 apresenta o diagrama de classes do agente.

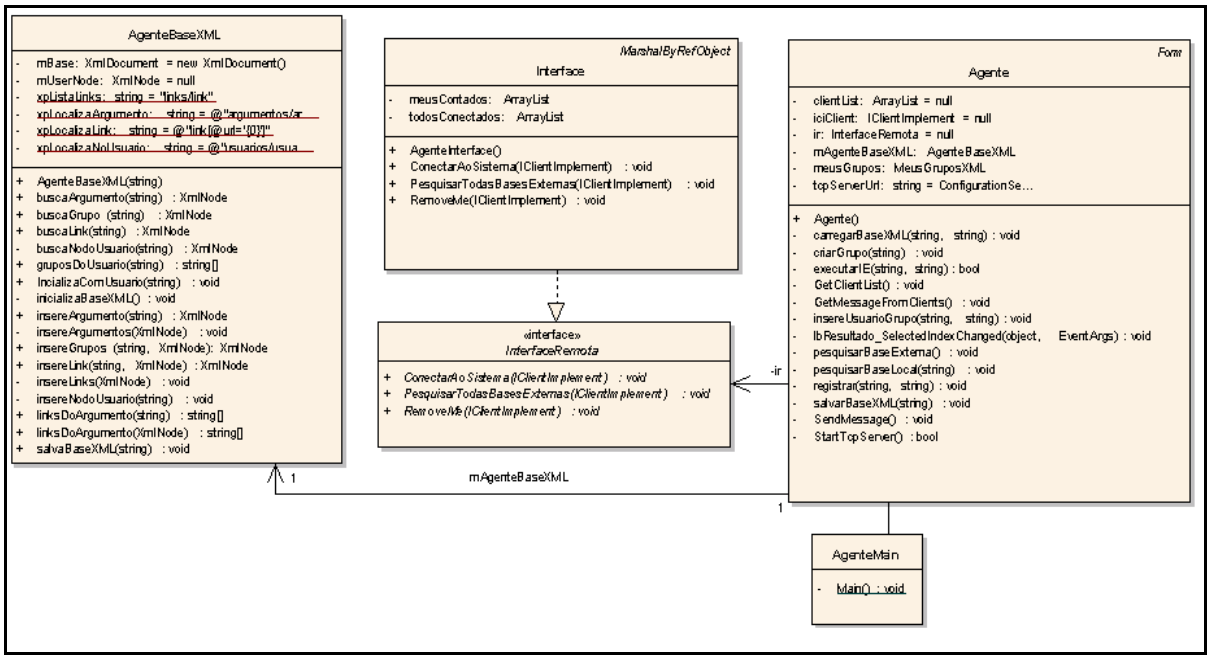


Figura 10 – Diagrama de classes do agente

A descrição do diagrama de classes do agente é apresentada da seguinte maneira:

a) a classe *Agente* contém os métodos necessários para realizar as buscas baseadas no argumento fornecido pelo usuário, inclui-se as buscas locais, em outros agentes e na Internet;
b) a classe *Interface* contém os métodos que serão usados tanto pelo cliente como pelo servidor;
c) a classe *AgenteBaseXML* contém os métodos e as definições para a manipulação da base de dados XML.

A Figura 11 apresenta o diagrama de classes do servidor, parte integrante do sistema que entre outras funções será responsável pela comunicação entre os agentes.

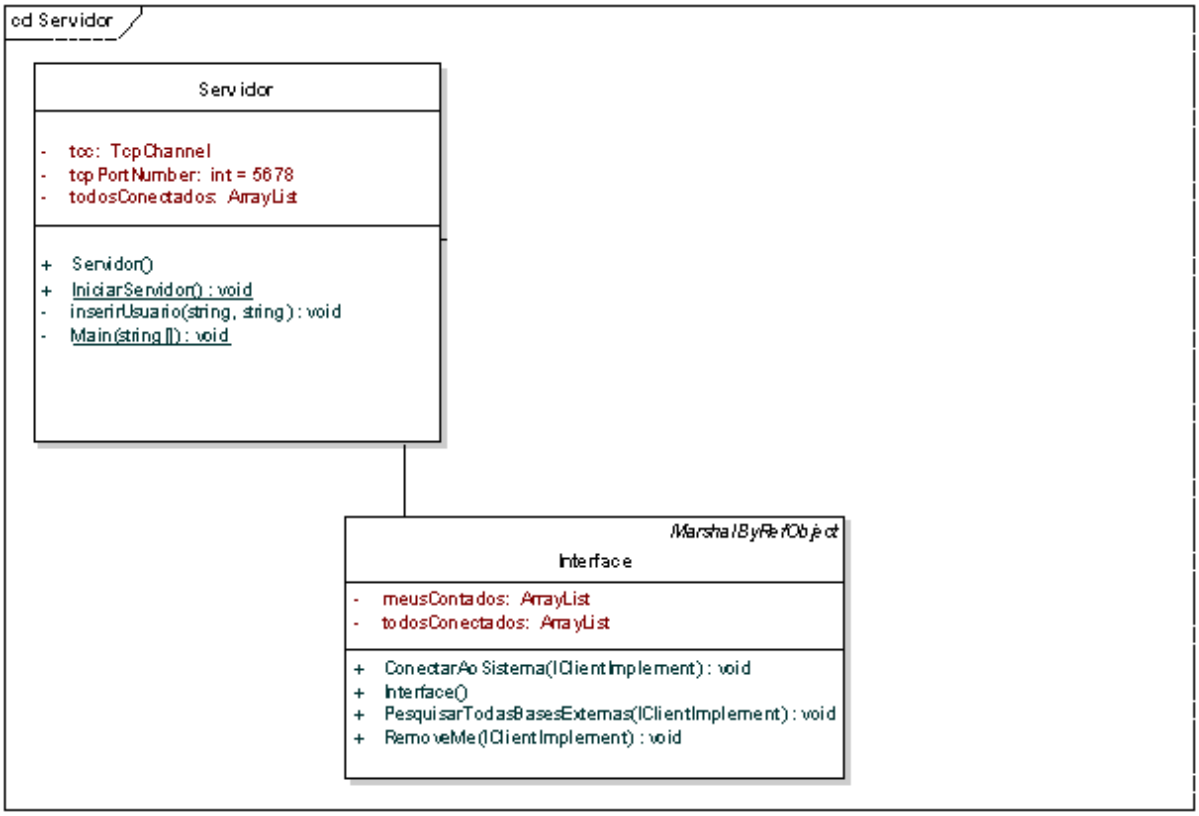


Figura 11 – Diagrama de classes do servidor

A descrição do diagrama de classes do servidor é definida da seguinte maneira:

a) a classe *Servidor* é responsável por receber as requisições dos clientes, atualizar a lista dos usuários conectados ao sistema e enviar essa lista aos mesmos;
b) o servidor é responsável também por implementar a *Interface*, para que os clientes utilizem os serviços providos por ela.

3.3.1.3 Diagrama de atividades

O diagrama de atividades descreve a seqüência das atividades de um sistema, elas apresentam comportamento condicional e por meio dessas condições apresentarão mudanças de estado (BOOCH; RUMBAUGH; JACOBSON, 2000, p. 257).

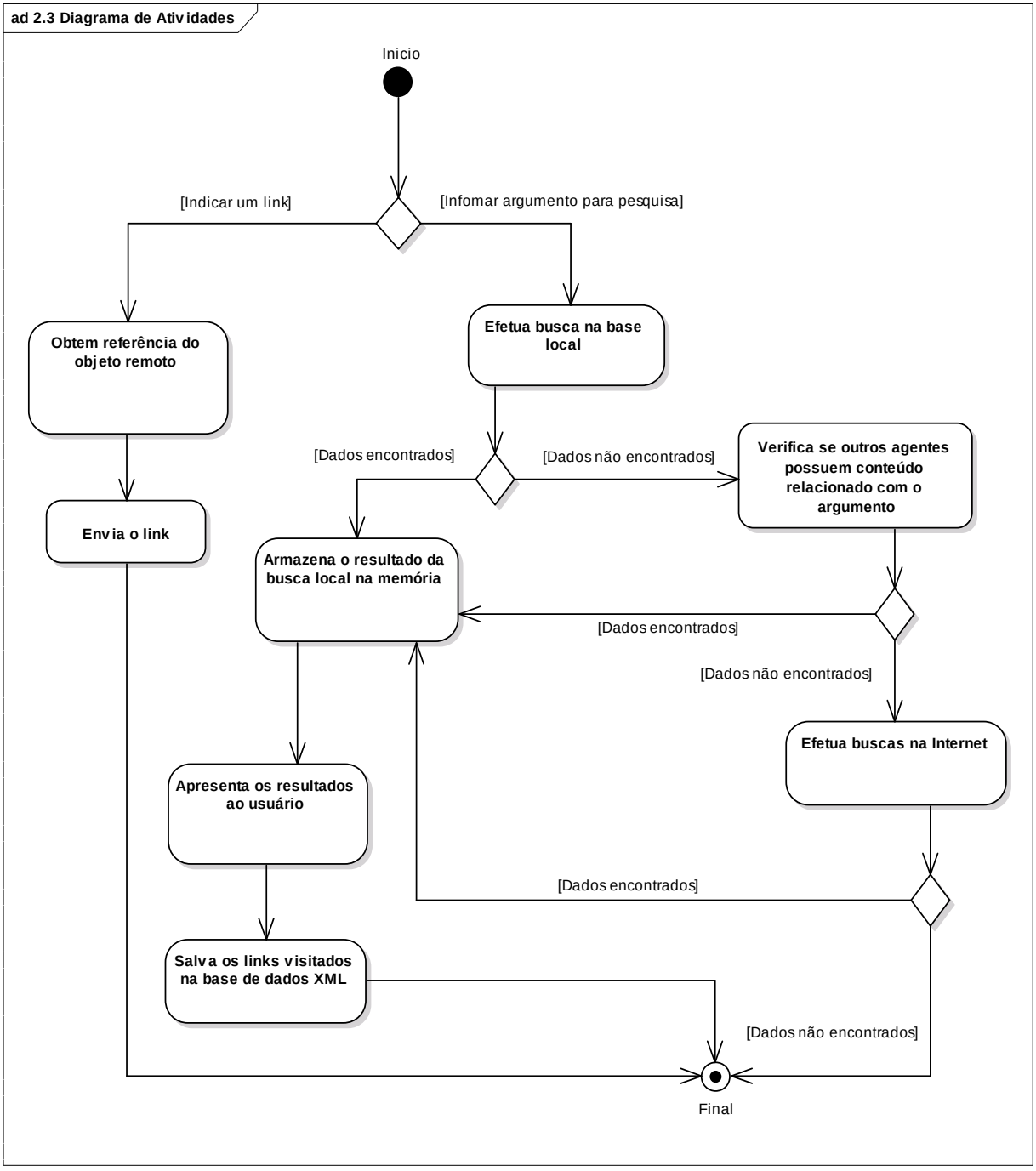


Figura 12 – Diagrama de atividades

A Figura 12 apresenta o diagrama de atividades do sistema, permitindo a visualização seqüencial da execução do sistema. A descrição do diagrama de atividades apresenta-se da seguinte maneira:

a) se o usuário optar por informar um link da Internet para outro usuário, o sistema obtém a referência do objeto remoto, ou seja, estabelece uma comunicação e envia o link;
b) se o usuário optar por informar um argumento para pesquisa, o sistema primeiramente efetuará uma busca na base de dados local;
c) se existir conteúdo relacionado com o argumento da pesquisa na base de dados local, o sistema armazenará em memória e em seguida efetuará uma consulta aos demais agentes do sistema;
d) se não existir conteúdo relacionado com o argumento da pesquisa na base de dados local, o sistema efetuará uma consulta aos demais agentes do sistema;
e) se os agentes apresentarem conteúdo relacionado com o argumento da pesquisa, o sistema armazenará em memória e em seguida efetuará uma consulta na Internet;
f) se os agentes não apresentarem conteúdo relacionado com o argumento da pesquisa, o sistema efetuará uma consulta na Internet;
g) se existir conteúdo relacionado com o argumento da pesquisa na Internet, o sistema armazenará em memória;
h) se não existir conteúdo relacionado com o argumento da pesquisa na Internet, o sistema verifica se possui dados em memória;
i) se existirem dados em memória o sistema apresentará ao usuário;
j) as páginas da Internet visitadas pelo usuário serão armazenadas na base de dados local.

3.3.2 Modelagem de dados

Para o armazenamento das informações optou-se pela utilização dos arquivos XML caracterizados pela portabilidade e facilidade no seu entendimento.

A Figura 13 apresenta o arquivo XML responsável pelo armazenamento dos grupos de contatos criados pelo usuário.

```
<MeusGrupos>
  <Grupo nome="Amigos">
    <usuarios>
      <usuario nome="Usuario" />
      <usuario nome="Usuariol" />
    </usuarios>
  </Grupo>
  <Grupo nome="Informatica">
    <usuarios>
      <usuario nome="Jandir Beppler" />
      <usuario nome="Usuario" />
      <usuario nome="Usuariol" />
    </usuarios>
  </Grupo>
</MeusGrupos>
```

Figura 13 – Arquivo XML para armazenamento dos grupos de contados

Na seqüência a apresenta-se um exemplo da base de dados de um agente, representado pela Figura 14. O arquivo XML que constitui a base de dados do agente armazenará o nome do usuário, os assuntos pesquisados bem como as respectivas páginas visitadas pelo usuário. A descrição do arquivo apresenta-se da seguinte maneira:

a) inicialmente cria-se o elemento *usuários* que conterá os elementos *usuário*;
b) na seqüência cria-se o elemento *usuário* que conterá os elementos *argumentos* e os atributos *nome*;
c) na seqüência cria-se o elemento *argumentos* que conterá os elementos *argumento;*
d) na seqüência cria-se o elemento *argumento* que conterá os elementos *links* e os atributos *args;*
e) na seqüência cria-se o elemento *links* que conterá os elementos *link;*
f) por fim cria-se o elemento *link* que conterá os atributos *url*.

A Figura 14 apresenta como os dados ficarão dispostos no arquivo XML.

```
<usuarios>
  <usuario nome="Jandir Beppler">
    <argumentos>
      <argumento args="Windows 2003">
        <links>
          <link url="http://www.microsoft.com/Windowsserver2003/" />
          <link url="http://www.microsoft.com/Windows/" />
        </links>
      </argumento>
      <argumento args="Windows XP">
        <links>
          <link url="http://www.microsoft.com/Windowsxp/" />
          <link url="http://www.microsoft.com/" />
        </links>
      </argumento>
    </argumentos>
  </usuario>
</usuarios>
```

Figura 14 – Organização dos dados no arquivo XML

3.4 IMPLEMENTAÇÃO

A implementação do sistema está divida em duas seções. Na primeira seção, apresentam-se algumas considerações sobre as técnicas e ferramentas utilizadas para a sua implementação. Na segunda seção, apresenta-se a operacionalidade ou o funcionamento da implementação.

3.4.1 Técnicas e ferramentas utilizadas

Para o desenvolvimento do presente trabalho foram utilizadas técnicas e ferramentas como a utilização de um *Web Service* para realizar as buscas na Internet, armazenamento de dados em arquivos XML, utilização da tecnologia .Net Remoting para a comunicação entre os agentes. Os detalhes de cada uma dessas técnicas ou ferramentas estão descritos a seguir.

No desenvolvimento de um assistente de informação ou de Internet, é imprescindível que se tenha um mecanismo eficiente para efetuar buscas na WWW. Após inúmeras pesquisas em busca de uma ferramenta adequada, optou-se por utilizar uma ferramenta fornecida pelo que acredita-se ser o meio mais eficiente de buscas de informações presentes na Internet, o *Google*. Essa ferramenta denomina-se *GoogleSearchService*, através do seu uso é possível usufruir das mesmas características apresentadas pela página de busca presente na Internet. O *GoogleSearchService* é um *Web Service*, ou seja, ele é um componente ou unidade de software que fornece uma funcionalidade específica, nesse caso é a busca de informações. Ele permite que diferentes sistemas possam acessá-lo e utilizar as suas funcionalidades. A Figura 15 apresenta um pequeno trecho de código utilizado para a implementação.

```
private void searchButton_Click(object sender, System.EventArgs e)
{
    lbResultado.Items.Clear();
    this.Cursor = Cursors.WaitCursor;

    // Criação de um objeto Google Search
    Google.GoogleSearchService s = new Google.GoogleSearchService();
    try
    {
        // Chamada do método de busca
        Google.GoogleSearchResult r = s.doGoogleSearch(keyBox.Text, searchBox.Text,
            0, 10, false, "", false, "", "", "");

        // Extrai uma estimativa do número de páginas encontradas na Internet
        int estResults = r.estimatedTotalResultsCount;

        //Apresenta os links da web
        for (int i = 0; i < (estResults > 10 ? 10 : estResults); i++)
        {
            lbResultado.Items.Add(r.resultElements[i].URL);
        }
        searchResultLabel.Text = Convert.ToString(estResults);
    }
    catch (System.Web.Services.Protocols.SoapException ex)
    {
        MessageBox.Show(ex.Message);
    }
    this.Cursor = Cursors.Default;
}
```

Figura 15 – Implementação do mecanismo de busca na Internet

Para o desenvolvimento da estrutura de armazenamento dos dados do sistema, utilizaram-se arquivos XML, os quais foram elaborados na plataforma .Net utilizando a linguagem C#. Essa plataforma de desenvolvimento oferece vários recursos para manipulação desse tipo de arquivos. A Figura 16 apresenta um pequeno trecho de código utilizado para a implementação. Nele tem-se a definição da função *insereNodoUsuario*, responsável por inserir o nome dos usuários aninhados com os argumentos de pesquisa que são inseridos pela função *insereArgumentos*.

```
private void insereNodoUsuario(string usuario)
{
    XmlNode root = mBase.DocumentElement;

    mUserNode = mBase.CreateElement("", "usuario", "");
    mUserNode.Attributes.Append(mBase.CreateAttribute("nome"));
    mUserNode.Attributes["nome"].Value = usuario;
    root.AppendChild(mUserNode);
    insereArgumentos(mUserNode);
}

private void insereArgumentos(XmlNode node)
{
    node.AppendChild(mBase.CreateElement("argumentos"));
}

public XmlNode insereArgumento(string argumento)
{
    XmlNode argumentos = mUserNode.SelectSingleNode("argumentos");
    XmlNode xArgumento = argumentos.AppendChild(mBase.CreateElement("argumento"));
    xArgumento.Attributes.Append(mBase.CreateAttribute("args"));
    xArgumento.Attributes["args"].Value = argumento;
    insereLinks(xArgumento);
    return xArgumento;
}
```

Figura 16 – Implementação da estrutura XML

Para o desenvolvimento da estrutura de comunicação entre os agentes utilizou-se o .Net Remoting, tecnologia presente na linguagem C#, para permitir a comunicação entre sistemas distribuídos. A Figura 17 apresenta um pequeno trecho de código utilizado para a implementação da comunicação entre os agentes. Todas as conexões realizadas pelos agentes se utilizarão da estrutura abaixo apresentada.

```
class ServidorRemoto
{
    private TcpChannel tcc;
    private int tcpPortNumber = 5678;
    public ServidorRemoto()
    {
        // Registrando o canal TCP
        BinaryClientFormatterSinkProvider bcfs = null;
        BinaryServerFormatterSinkProvider bsfs = new BinaryServerFormatterSinkProvider();
        bsfs.TypeFilterLevel = TypeFilterLevel.Full;
        id = new Hashtable();
        id["port"] = tcpPortNumber;
        id["typeFilterLevel"] = TypeFilterLevel.Full;
        id["name"] = System.Guid.NewGuid().ToString();
        tcc = new TcpChannel(id, bcfs, bsfs);
        ChannelServices.RegisterChannel(tcc);
        id.Clear();
        id = null;
    }
    public static void iniciarServidor()
    {
        try
        {
            RemotingConfiguration.RegisterWellKnownServiceType(typeof(AgenteInterface), "Servidor.soap",
                WellKnownObjectMode.Singleton);
        }
        catch (RemotingException ex)
        {
            Console.WriteLine(ex.Message);
            throw new UserExceptions(ex.Message, ex);
        }
    }
```

Figura 17 – Implementação do .Net Remoting

A classe *ServidorRemoto* apresentada pela Figura 17, efetua o registro do canal TCP que será utilizado para a comunicação entre os agentes. Na seqüência a Figura 18 apresenta um trecho de código que implementa a conexão dos usuários ao sistema.

```
public void ConectarAoSistema(IClientImplement ici)
{
    if (ici != null)
    {
        try
        {
            // Verifica se o cliente já está conectado
            foreach (IClientImplement icc in this.todosConectados)
            {
                if (icc.UserName.CompareTo(ici.UserName) == 0)
                {
                    throw new UserExceptions(errorMessages[0]);
                }
            }
            this.todosConectados.Add(ici);
            Console.WriteLine("--------------------Entrada de novo usuario----------------");
            Console.WriteLine(ici.UserName + " Conectou em : " + DateTime.Now.ToString());
            Console.WriteLine("-----------------------------------------------------------");

            //Carrega o Arraylist ClientList
            foreach (IClientImplement icimp in this.todosConectados)
            {
                ici.ClientList = this.todosConectados;
            }
        }
        catch (RemotingException ex)
        {
            Console.WriteLine(ex.Message);
            throw new UserExceptions(ex.Message, ex);
        }
    }
}
```

Figura 18 – Conexão com o sistema

Quando um usuário realizar uma pesquisa, além da Internet, as outras fontes de dados a serem consultadas serão a base de dados local e a base de dados dos outros agentes pertencentes ao sistema. Na seqüência apresentam-se trechos de código utilizados para o desenvolvimento das rotinas de pesquisa.

```
private void pesquisarBaseLocal(string argumento)
{
    carregarBaseXML(tbNMArquivo.Text, tbNome.Text);
    string[] links = mAgenteBaseXML.linksDoArgumento(argumento);
    lbResultado.Items.Clear();
    if (links != null)
        foreach (string link in links)
        {
            lbResultado.Items.Add("Link local: " + link);
        }
}
```

Figura 19 – Pesquisa na base de dados local

A primeira fonte de dados a ser consultada pelo agente será a base de dados local representada pela Figura 19. A função *pesquisarBaseLocal*, irá percorrer a base de dados do agente em busca de informações relacionadas com o argumento da pesquisa informado pelo usuário. Após a pesquisa local o agente fará uma consulta aos outros agentes do sistema para verificar se eles possuem algum conteúdo relacionado com o argumento em questão.

```
private void enviarMsgExterna()
{
    if (this.searchBox.Text.CompareTo(string.Empty) != 0)
    {
        this.ce.Message = searchBox.Text.Trim();
        ce.ReceiverName = "ALL";
        this.ir.PesquisarTodasBasesExternas(ce);
        this.searchBox.Text = string.Empty;
    }
}


private void pesquisarBaseExterna()
{
    Thread th = new Thread(new ThreadStart(this.enviarMsgExterna));
    th.Start();
}
```

Figura 20 – Consulta aos agentes do sistema

A comunicação entre os agentes apresentada pela Figura 20 inicia-se quando o usuário informa um argumento para pesquisa, após o agente fazer uma busca na base de dados local, ele passa o argumento da pesquisa para os demais agentes do sistema através do método

this.ce.Message presente na função *enviarMsgExterna* essa função contém ainda o método *ce.ReceiverName* responsável por identificar se a consulta será enviada para todos os usuários, para um grupo de usuários previamente definido ou apenas para um único usuário especificado. A função *pesquisarBaseExterna* inicia uma *Thread* e chama a função *enviarMsgExterna* nesse momento é disparado a solicitação ao agentes que estiverem conectados.

3.4.2 Operacionalidade da implementação

Inicialmente o agente não estará conectado ao sistema, ou seja, é necessário que o usuário o faça explicitamente, a partir desse momento o agente estará apto a enviar e receber consultas referentes ao argumento informado pelo usuário. A Figura 21 apresenta o servidor em execução, aguardando a conexão dos agentes.

Figura 21 – Servidor em execução

A partir do momento em que o servidor estiver em execução, o usuário poderá realizar a conexão, como pode ser observado na Figura 22.

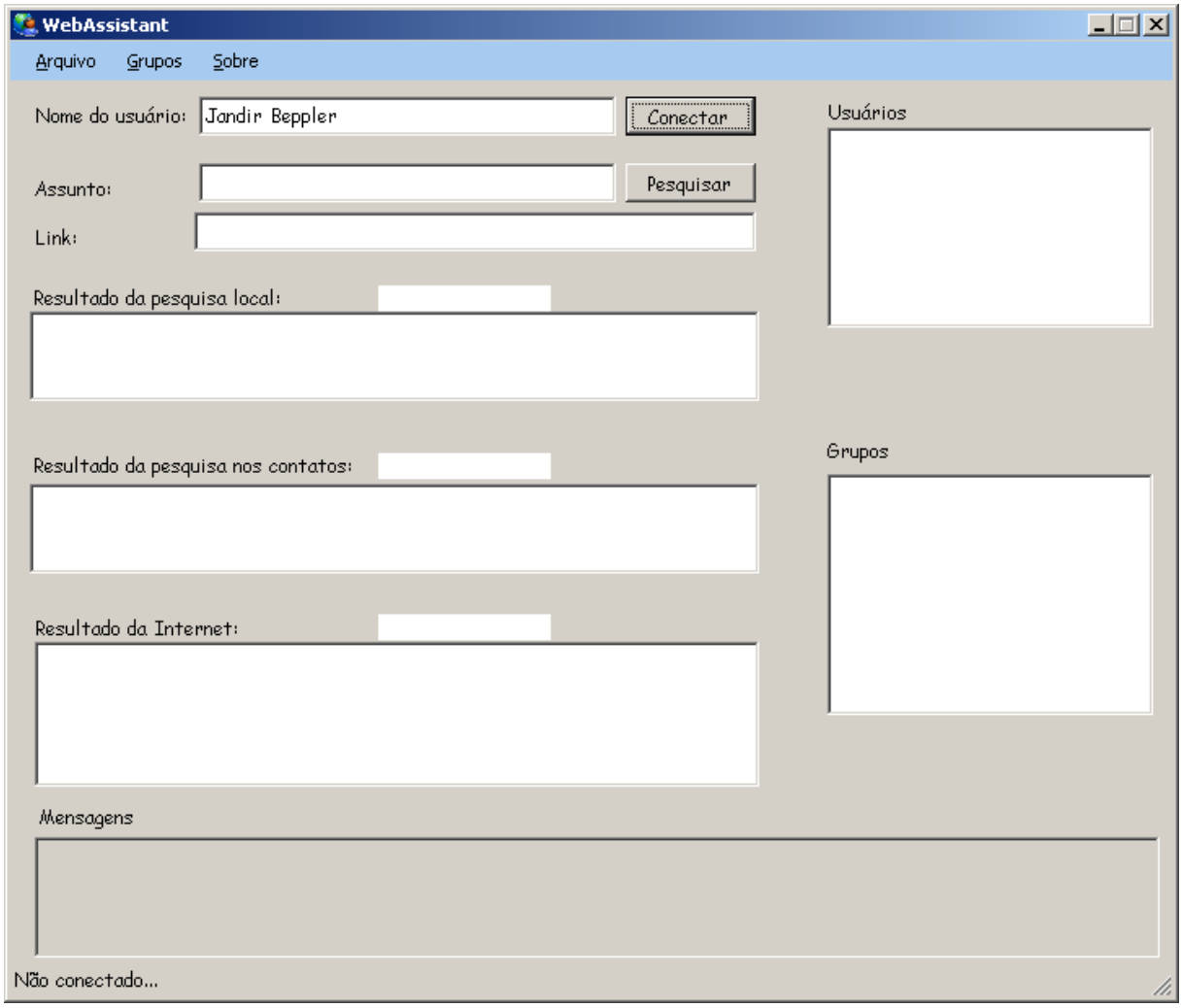


Figura 22 – Informações para conexão

Quando o usuário clicar no botão *Conectar*, as informações de conexão são enviadas ao servidor, que verifica se a identificação do usuário é única, ou seja, é permitida apenas uma conexão por usuário no servidor. Após essa verificação, o agente do usuário em questão poderá trocar informações com os demais agentes conectados.

Após os procedimentos de conexão apresentados anteriormente, o usuário poderá então realizar suas pesquisas. O sistema pode ser visto em funcionamento através da descrição do estudo de caso apresentado a seguir.

Imagina-se a necessidade de um usuário realizar uma busca na internet referente ao assunto “Windows 2003”. A Figura 23 ilustra essa ação apresentando os resultados obtidos através da realização das buscas.

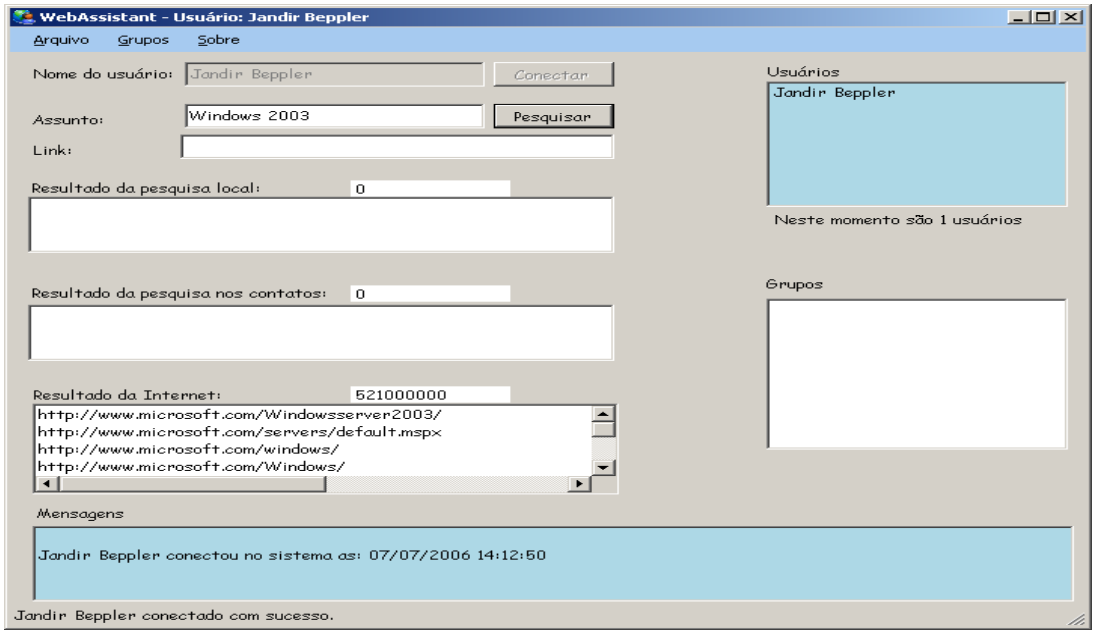


Figura 23 – Sistema em execução pela primeira vez

Observa-se que nesse caso o sistema está sendo executado pela primeira vez, logo a base de dados do agente está vazia fazendo com que a busca na base local não retorne nenhum valor. Observa-se também que há apenas um usuário conectado ao sistema, dessa forma não haverá troca de informações com outros agentes.

Na seqüência, a Figura 24 ilustra os resultados obtidos através da realização de uma segunda busca utilizando o mesmo argumento de pesquisa.

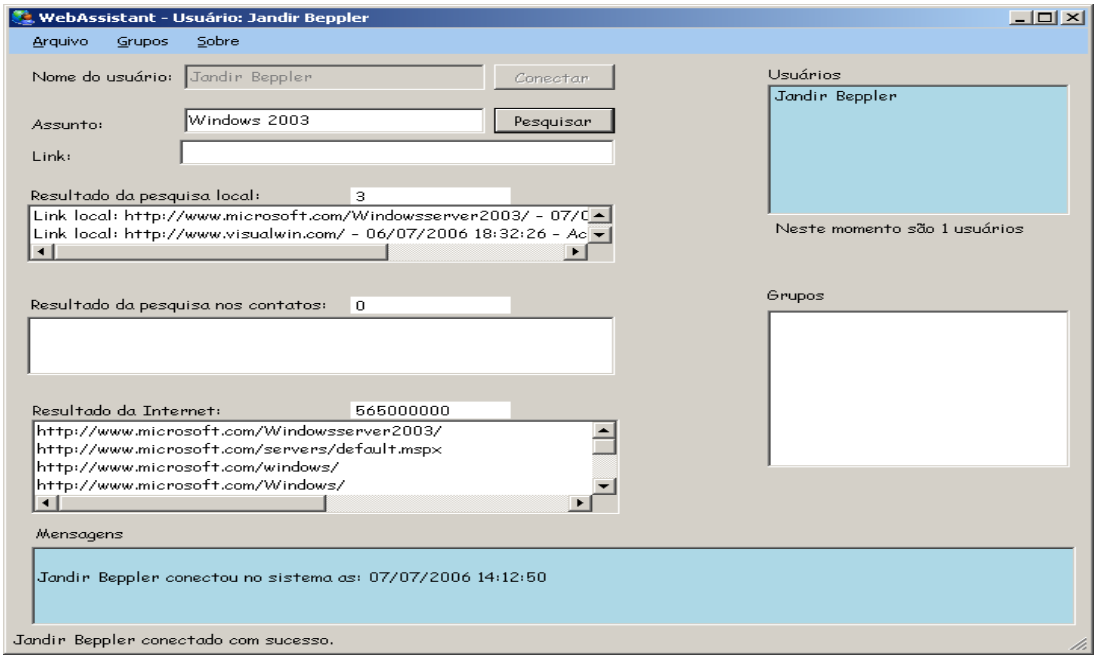


Figura 24 – Sistema em execução pela segunda vez

Observa-se que a Figura 24 apresenta alguns resultados precedidos pela expressão “Link local”, que significa que uma busca com o mesmo argumento já havia sido realizada e o usuário acessou as páginas da Internet em questão.

Na seqüência, a Figura 25 ilustra os resultados obtidos através da realização de uma terceira busca utilizando o mesmo argumento de pesquisa.

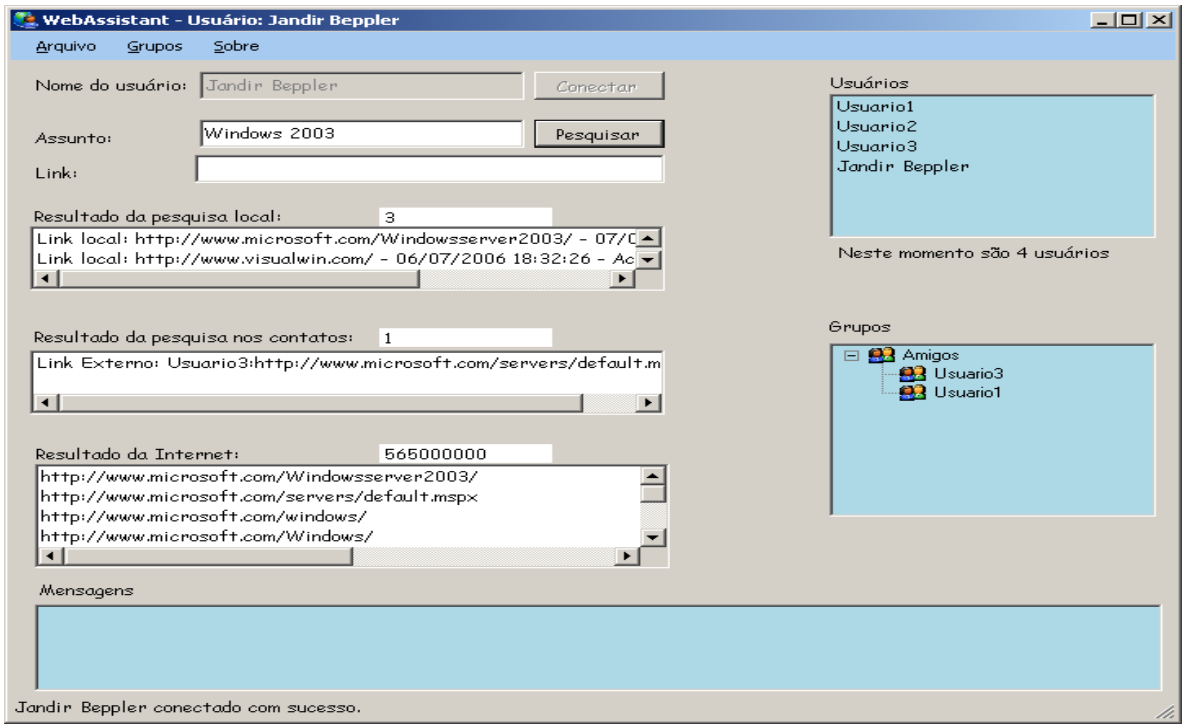


Figura 25 – Sistema em execução pela terceira vez

Observa-se que a Figura 25 apresenta os três tipos de resultados possíveis, os vindos da Internet, os resultados obtidos em pesquisas anteriores precedidos pela expressão “Link local”, e por fim os resultados encontrados na base de dados dos outros agentes do sistema precedidos pela expressão “Link Externo”.

O estudo de caso anteriormente apresentado evidencia a importância do “trabalho” efetuado pelo agente durante as pesquisas realizadas pelo usuário. Observa-se a possibilidade da criação de grupos de usuários, permitindo a busca em agentes de usuários de um grupo específico.

3.5 RESULTADOS E DISCUSSÃO

Após várias execuções do sistema observou-se que o mesmo apresentou excelentes resultados no que diz respeito à localização das informações desejadas. Destacando-se principalmente, a rede de contatos formada pelos usuários juntamente com seus respectivos agentes. Dessa forma o sistema se caracteriza como sendo multi-agentes, permitindo que esses troquem informações entre si tornando o sistema mais eficaz a cada nova consulta realizada, uma vez que a base de dados de cada agente é alimentada constantemente.

Fazendo um comparativo com a ferramenta GoogleDesktop (GOOGLE INC, 2005), percebe-se que elas obtém resultados bem semelhantes. A diferença se restringe às funcionalidades, ou seja, enquanto o presente trabalho se propõe a aperfeiçoar as pesquisas por informações baseadas em argumentos fornecidos pelos usuários, o GoogleDesktop possibilita a obtenção de informações sobre o mercado financeiro, meteorologia e também visualizar as principais notícias em tempo real entre outras.

4 CONCLUSÕES

Este trabalho apresentou uma ferramenta para auxiliar os usuários de microcomputadores no processo de busca de informações da Internet. O sistema mostrou-se eficaz no atendimento do seu propósito, proporcionando uma alternativa viável para a diminuição do tempo gasto para que as informações desejadas sejam encontradas.

Dentre as principais vantagens do sistema desenvolvido, destaca-se o emprego de técnicas de sistemas distribuídos na implementação do módulo de comunicação do sistema, e também a utilização da plataforma de desenvolvimento .Net utilizando a linguagem C#. Tais características permitiram o desenvolvimento do trabalho utilizando tecnologias recentes no mercado, exigindo um esforço extra para o seu entendimento, mas proporcionando o resultado esperado. A desvantagem que se pode citar é que o sistema deveria agir com mais autonomia, ou seja, ele deveria criar um perfil do tipo de informação que o usuário pesquisa, para poder no futuro apresentar resultados mais precisos em relação ao tema da pesquisa informado.

4.1 EXTENSÕES

Este trabalho pode ser continuado através da implementação de mecanismos de inferências para o agente, ou seja, desenvolver rotinas para permitir que o agente identifique o conteúdo das páginas da Internet encontradas com o argumento da pesquisa. Dessa forma o agente se tornará mais “inteligente”, pois uma vez que ele identifica o conteúdo dos documentos ele pode criar um perfil para as buscas do usuário, permitindo que ele haja autonomamente em futuras buscas que o usuário efetuar. Também sugere-se a implementação de buscas em paralelo, ou seja, implementar um módulo de comunicação em cada agente para permitir que a troca de informações tenha continuidade mesmo em caso de queda do servidor.

REFERÊNCIAS BIBLIOGRÁFICAS

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML:** guia do usuário. Tradução Fábio Freitas. Rio de Janeiro: Campus, 2000.

COPERNIC TECHNOLOGIES INC. **Copernic Agent**. [S.l.], 2006. Disponível em: <http://www.copernic.com/index.html>. Acesso em: 27 maio 2006.

COSTA, M. T. C. **Uma arquitetura baseada em agentes para suporte ao ensino à distância.** 1999. Tese (Doutorado em Engenharia de Produção) - Curso de Pós-Graduação em Engenharia de Produção, Universidade Federal de Santa Catarina, Florianópolis. Disponível em: <http://www.eps.ufsc.br/teses99/thiry>. Acesso em: 15 maio 2006.

DEITEL, H. M. et al. **C# como programar**. Tradução João Eduardo Nóbrega Tortello. São Paulo: Pearson Education, 2003.

DIAS, P. **Utilizando XML para troca de informações entre empresas**. 2004. 58 f. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) – Instituto Superior Tupy, Joinville. Disponível em: <http://www.laureano.eti.br/ensino/orientacoes/ist_2004_patricia_xml.pdf>. Acesso em: 27 maio 2006.

IBM CORPORATION. **Policy-based system management:** collaboration and cooperation. [S.l.], 2003. Disponível em: <http://www.almaden.ibm.com/asr/chiacs/presentations/kandogan.pdf>. Acesso em: 29 maio 2006.

GOOGLE INC. **GoogleDesktop**. Mountain View, California, 2005. Disponível em: <http://desktop.google.com/>. Acesso em: 27 maio 2006.

LOBATO, C. A.; DAMASCENO, K. N. F. **Agentes móveis:** Aglets na busca de informações. 2003. 192 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Universidade Federal do Pará, Belém. Disponível em: <http://www.cultura.ufpa.br/informatica/tcc/karla_cidiane.pdf>. Acesso em: 27 maio 2006.

LOTH, C. A. **Tecnologias distribuídas no .NET Framework**. Porto Alegre, 2005. Disponível em: < www.inf.ufrgs.br/~asc/sodr/pdf/RemotingDotNet.ppt>. Acesso em: 29 maio 2006.

MAIA, R. F. **Sistema multi-agentes para acompanhamento e auxílio de avaliação de alunos em ambientes de ensino à distância**. 2004. 189 f. Dissertação (Mestrado em Engenharia Elétrica) – Escola Politécnica da Universidade de São Paulo, São Paulo. Disponível em: <http://cognitio.incubadora.fapesp.br/portal/producao/teses%20e%20dissertacoes/rodrigo%20maia/2004.12_RFM_DissertacaoMestrado-VFR.pdf >. Acesso em: 27 maio 2006.

MARCHAL, B. **XML conceitos e aplicações**. Tradução Daniel Vieira. São Paulo: Berkeley, 2000.

MICROSOFT CORPORATION. **Escolhendo entre WebServices, Enterprise Services e Remoting.** [S.l.], 2006. Disponível em: <http://www.microsoft.com/brasil/msdn/Tecnologias/arquitetura/Escolhendo.mspx>. Acesso em: 04 maio 2006.

MICROSOFT CORPORATION. **MSN Search.** [S.l.], 2005. Disponível em: < http://www.msn.com/ >. Acesso em: 27 maio 2006.

PAULI, G. **.NET Remoting:** criando um chat usando objetos distribuídos. Rio de Janeiro, 2004. Disponível em: <http://www.neoficio.com.br/msdn/colunistas/guinther/01_ChatNET.asp>. Acesso em: 29 maio 2006.

ROBINSON, S. et al. **Professional C#.** Birmingham: Wrox, 2001.

SILVA, H. P.; THIRY, M.; ABREU, A. F. Monitoramento automatizado na internet: uma resposta ao desafio de melhores serviços a custos baixos para as bibliotecas universitárias. In: SEMINÁRIO NACIONAL DE BIBLIOTECAS UNIVERSITÁRIAS, 11., 2000, Florianópolis. **Anais...** Florianópolis: UFSC, 2000. Disponível em: <http://snbu.bvs.br/snbu2000/docs/pt/doc/t001.doc>. Acesso em: 18 maio 2006.

SILVA, G. B. **Uma proposta de criação perfis de usuários da Internet para filtragem de informação personalizada**. 2002. 75 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Universidade Federal de Lavras, Lavras. Disponível em: <http://www.comp.ufla.br/monografias/ano2002/Uma_proposta_de_criacao_de_perfis_de_usuarios_da_internet_para_filtragem_de_informacao_personalizada.pdf>. Acesso em: 27 maio 2006.

VAHL, J. C. J. **Uso de agentes de interface para adequação de bate-papos ao contexto de educação à distância.** 2003. 151 f. Dissertação (Mestrado em Ciências da Computação) – Instituto de Computação, Universidade Estadual de Campinas, Campinas. Disponível em: <http://teleduc.nied.unicamp.br/pagina/publicacoes/zeh_disser.pdf>. Acesso em: 15 maio 2006.