Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Departamento de Sistemas e Computação

Relatório de Estágio

Avaliação do uso de .NET Remoting no BRSIL

Diego Renato dos Santos
diegords@lcc.ufcg.edu.br

Campina Grande
Outubro de 2007

Biblioteca Setorial do CDSA. Maio de 2021.

Sumé - PB

Diego Renato dos Santos

Avaliação do uso de .NET Remoting no BRSIL

Aprovado em: _____

Banca Examinadora:

_____

Prof. Ph. D. Péricles Rezende Barros

Supervisor acadêmico

_____

Prof. D. Sc. José Sérgio da Rocha Neto

Membro da banca

_____

Prof. M. Sc. Bruno Correia da Nóbrega Queiroz

Membro da banca

Apresentação

Este documento corresponde ao relatório final para o componente curricular de Estágio Integrado do Curso de Ciência da Computação, da Universidade Federal de Campina Grande.

O relatório descreve as atividades de pesquisa bibliográfica e desenvolvimento no projeto BRSIL e no projeto Wireless, no Laboratório de Instrumentação Eletrônica e Controle (LIEC), do Departamento de Engenharia Elétrica da Universidade Federal de Campina Grande. A atividade consistia em analisar a viabilidade do Uso de .NET Remoting, assim como a utilidade de implementar uma possível modificação do sistema com base nessa análise. Como meio-termo, também foi utilizada a implementação do .NET Remoting no projeto Wireless, também desenvolvido no LIEC. Ambos os projetos estão sendo desenvolvidos com a parceria da Petrobras e da FINEP.

Sumário

Glossário

Intrinsecamente seguro: diz-se dos equipamentos que não soltam faíscas elétricas em condições normais, como por exemplo uma queda, ou enquanto é ligado. Isso é importante em ambientes com riscos de explosão

LOPA: sigla para *Layer of Protection Analysis*, é o nome de uma técnica semiquantitativa para análise de riscos em instalações industriais.

Domínio de aplicação: na plataforma .NET, domínio de aplicação é uma estrutura de dados que se comporta como um processo, porém um mesmo processo do sistema operacional pode ter mais de um domínio de aplicação

IIS: Internet Information Services, o servidor web comum em ambientes Windows

SSL: Secure Socket Layer, é um protocolo para a utilização de comunicações seguras

SOAP: Simple Object Access Protocol, é um protocolo de serialização de objetos no formato XML para aplicações distribuídas.

Lista de figuras

Lista de tabelas

CAPÍTULO 1
INTRODUÇÃO

1. Introdução

As indústrias, dos mais diversas áreas de atividade estão em uma procura crescente pelo aumento da produtividade. Comumente, esse aumento é proporcionado através da utilização cada vez mais intensa de soluções tecnológicas em seus processos. Porém, a conjugação de equipamentos de diversos fabricantes para os mais variados objetivos em plantas industriais pode acarretar em novos riscos. Em muitos setores da indústria, esses riscos podem ser altos, tanto em termos materiais ou de mortes. Por exemplo, em indústrias petroquímicas ou semelhantes, pode haver risco de vazamentos de produtos tóxicos, explosões ou outros danos à planta, o ambiente ao seu redor e aos que nela estiverem presentes.

Para maximizar a segurança são necessários sistemas de segurança integrados, que possam mensurar os riscos para que a administração possa adotar medidas de segurança necessárias, mas que não traga custos adicionais desnecessários. Estes sistemas podem ser não-automáticos, semi-automáticos e automáticos. Dentre os sistemas de controle automático. Este último está voltado para a segurança do processo de forma a garantir que estas mesmas variáveis estejam dentro de limites considerados seguros para a operação da unidade.

Aliado aos equipamentos instalados para coleta de variáveis para obtenção de métricas relativas à segurança, é utilizado um conjunto de soluções de *software* para a análise automática dos dados colhidos para facilitar o julgamento do nível de segurança que deve ser adotado. Dentro dessa categoria de *software* se situa o BRSIL. Ela é uma ferramenta de análise do ciclo de vida de segurança de Funções Instrumentadas de Segurança.

O BRSIL possui um grande número de funções desenvolvidas, sendo que suas funcionalidades estão quase totalmente completas no modo local, ou seja, instalado em uma máquina. Mas planeja-se utilizá-lo também de forma remota, através de uma rede corporativa ou até pela Internet. Tendo em vista essas possibilidades, está ocorrendo um processo de pesquisa pelas possíveis tecnologias de comunicação entre processos podem ser usadas, inclusive a .NET Remoting, discutida neste documento.

CAPÍTULO 2
AMBIENTE DE ESTÁGIO

2. Ambiente de estágio

O aluno desenvolveu seu estágio no Laboratório de Instrumentação Eletrônica e Controle (LIEC) do Departamento de Engenharia Elétrica (DEE) da Universidade Federal de Campina Grande (UFCG).

Figura 1 - Vista externa do LIEC (e Embedded)

Localizado no prédio Gurdip Singh Deep (Bloco CI2), junto do Laboratório de Sistemas Embarcados e Computação Pervasiva (*Embedded*), do mesmo departamento, o LIEC conta com diversos alunos dos cursos de Ciência da Computação e Engenharia Elétrica, além de alunos de mestrado e doutorado, participando de diversos projetos de pesquisa nas áreas de instrumentação eletrônica, automação industrial, identificação de sistemas e controle de processos, através de parcerias com instituições públicas e privadas.

O LIEC conta com equipamentos, que permitem simular ambientes industriais e seus respectivos processos.

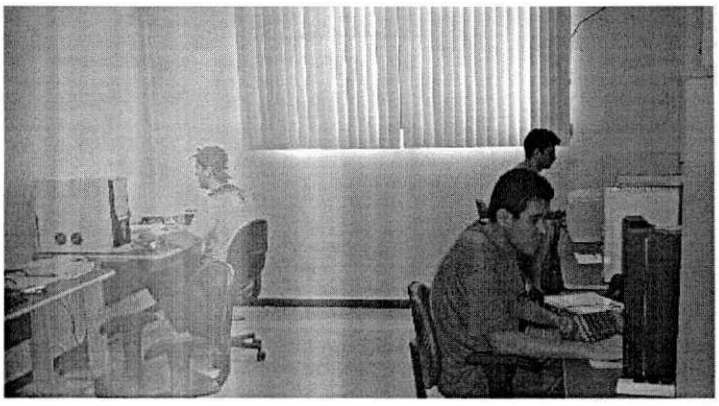

Figura 2 - Local de trabalho em uso.

A rede local do laboratório está baseada em *Ethernet* de um *gigabit* conectada à rede do campus da Universidade Federal de Campina Grande e à Internet. Além disso, todo o laboratório está equipado com acesso à Internet via rede sem fio (*Wi-Fi*). A biblioteca do laboratório está em constante atualização, contendo livros recentes adquiridos através de recursos de projetos.

2.1 Instrumentos de trabalho

O computador mais utilizado pelo aluno conta com processador AMD Athlon 64 3000+ , 2 gigabytes de memória principal. disco de 80 gigabytes serial ATA, e gravador de CD. O sistema operacional utilizado foi o Microsoft Windows XP Service Pack 2. Os softwares utilizados foram o Microsoft Visual Studio 2005 Professional Edition para o desenvolvimento do código, o Microsoft SQL Server 2005 como gerenciador de banco de dados e o Sparx Systems Enterprise Architect para a geração de diagramas UML.

Para o desenvolvimento em dispositivos móveis foram utilizados dois computadores de mão, um da série HP (Hewlett-Packard) iPAQ hx4700 para o desenvolvimento cotidiano, e outro da Ecom Instruments i.roc 420, sendo este último PDA intrinsecamente seguro.

2.2 Supervisão

A presença dos supervisores foi extremamente importante para freqüente consultas no caso de dúvidas.

2.2.1 Supervisor Acadêmico

Nome: Péricles Rezende Barros, Ph.D

E-mail: prbarros@dee.ufcg.edu.br

2.2.2 Supervisor Técnico

Nome: George Acioli Junior

E-mail: georgeacioli@yahoo.com.br

2.3 Cronograma de Execução

A execução deste projeto se divide entre as seguintes etapas a serem cumpridas como especificado abaixo:

- 1° Etapa – Estudo do contexto do projeto;
- 2° Etapa – Estudo da tecnologia .NET Remoting;
- 3° Etapa – Desenvolvimento de aplicações baseadas no .NET Remoting;
- 4° Etapa – Avaliação da utilização de .NET Remoting no contexto do projeto Confiabilidade de Sistemas Instrumentados de Segurança;
- 5ª Etapa – Escrita do relatório do estágio

Atividades	**Junho**				**Julho**				**Agosto**				**Setembro**			
1ª Etapa																
2ª Etapa																
3ª Etapa																
4ª Etapa																
5ª Etapa																

Tabela 1 – Cronograma do Estágio

CAPÍTULO 3
CONCEITOS BÁSICOS

3. Conceitos básicos

3.1 Sistemas instrumentados de segurança

Um Sistema Instrumentado de Segurança (*Safety Instrumented Systems* - SIS) é um sistema eletrônico projetado para responder a condições anormais de processo prevenindo o perigo ou reduzindo as conseqüências potenciais desse perigo. Um SIS monitora continuamente variáveis selecionadas, mas permanece inativo até que uma condição anormal e possivelmente perigosa ocorra.

Algumas características dos SIS são:

- Levar automaticamente um processo industrial para um estado seguro quando condições específicas forem violadas;
- Permitir que o processo seja executado normalmente quando condições específicas permitirem;
- Executar ações que reduzem as conseqüências de um acidente industrial.

Uma Função Instrumentada de Segurança (*Instrumented Safety Function* - SIF) é uma ação adotada por um SIS para levar o processo ou equipamento a um estado seguro. Esta função é um único equipamento que protege contra um perigo específico. O SIS, por outro lado, é um conjunto de SIFs, implantadas para uma unidade de processo ou uma planta.

Na década de 90, empresas e grupos industriais desenvolveram normas para projetar, construir e manter um Sistema Instrumentado de Segurança. Um dado de entrada chave para as ferramentas e técnicas necessárias para implementar estas normas era a probabilidade de falha exigida para cada SIF.

Em 1996, a ISA (The *Instrumentation, Systems, and Automation Society*) publicou uma norma para guiar a classificação de Sistemas Instrumentados de Segurança para indústrias de processo dos Estados Unidos, a norma ANSI/ISA-S84.01, que introduziu o conceito de Nível de Integridade de Segurança (*Safety Integrity Level* – SIL). Subseqüentemente, em 1998, o IEC (*International Eletrotechnical Commission*), com sede em Genebra, começou a elaborar a norma de segurança IEC61508 para auxiliar as empresas que utilizam sistemas instrumentados de segurança a proteger seu pessoal e suas instalações de eventos perigosos. A norma, formalmente intitulada "Segurança Funcional de Sistemas de Segurança Elétricos/Eletrônicos/Eletrônico-

Programáveis", é composta de sete partes que orientam o adequado gerenciamento do ciclo de vida e de todos os componentes do SIS.

Probabilidades de Falha na Demanda (*Probability of Failure on Demand* - PFD) aceitáveis para casa SIF precisam ser determinadas para o projeto e posterior verificação. O SIL é a representação estatística da disponibilidade de uma SIF quando uma demanda de processo ocorre, sendo utilizada em ambas as normas ANSI/ISA-S84.01 e IEC 61508. Quanto maior a importância da segurança em relação ao processo, maior deve ser a disponibilidade da função de segurança e, portanto, maior será o SIL

A IEC 61508 define quatro SILs para representar estatisticamente a integridade do SIS quando ocorre uma demanda do processo. O SIL leva em conta a integridade do dispositivo, a arquitetura, os diagnósticos, as falhas sistemáticas e aquelas por causas comuns, testes, operação e manutenção.

Um SIL estabelece a ordem de grandeza para a meta de redução de risco. Esta meta de magnitude de falha é a probabilidade almejada para falhas perigosas em relação aos requisitos de integridade da segurança. Esta grandeza pode ser especificada de duas formas distintas, dependendo do modo de operação do processo: em termos de falha em executar a função de segurança em demanda (para operação em baixa demanda) ou em probabilidade de falha perigosa por hora (para alta demanda ou operação em modo contínuo). Quanto maior for o número do SIL, maior será o impacto de uma falha e menor será a taxa de falhas aceitável.

Existem diversos enfoques para a determinação do SIL requerido por um sistema de segurança. Dentre eles, alguns métodos qualitativos - como a Matriz Tridimensional de Risco ou o Gráfico de Risco - são de emprego mais diretos e rápidos. A LOPA (*Layer of Protection Analysis* - Análise de Camadas de Proteção) é uma metodologia semiquantitativa que envolve as etapas de identificação de perigos, determinação de freqüência de eventos iniciadores, definição de critérios de aceitabilidade de riscos e, finalmente, para cada evento iniciador, a análise de cada camada de proteção de perigo para determinar se os critérios de aceitabilidade de risco estão sendo alcançados. Caso não estejam, será necessário incluir uma camada de proteção adicional (como uma SIF) ou fortalecer uma camada existente.

A experiência mostra que técnicas diferentes podem acarretar em resultados diferentes na hora de selecionar o SIL. Isto ocorre devido à dificuldade destes métodos em considerar o critério de risco. Métodos quantitativos, além de considerarem critérios de aceitabilidade de risco, possuem uma visão mais ampla do problema, não

sobrecarregando no SIS fatores de redução de riscos que poderiam estar em camadas cuja implementação é mais barata que o custo de implantação de uma SIF.

O SIS superestimado acaba encarecendo desnecessariamente a solução a ser implantada, sem nenhum benefício adicional, enquanto um SIS subestimado pode comprometer a segurança de toda unidade industrial, por tratar-se de uma camada de proteção e mitigação de riscos.

3.2 .NET Remoting

.NET Remoting provê um nível de indireção na manipulação de objetos de forma transparente. A título de exemplo, supondo-se um sistema em que há dois processos: um processo "servidor" e outro processo "cliente". É possível manipular objetos alocados no processo servidor (objetos remotos) de modo tão simples como se estivessem no processo cliente.Para que o objeto do servidor possa ser visto como um objeto remoto pelos clientes, é necessário que ele obedeça determinadas características. Ele devem permitir uma operação chamada *marshaling*, que permite que esse objeto possa ser manipulado de alguma maneira que possibilite este transpor seu domínio de aplicação.

Existem dois tipos de *marshaling*:

- **Por valor:** os valores contidos no objeto do servidor são serializados e copiados para o outro domínio de aplicação. Para que isso possa acontecer, é necessário que eles sejam serializáveis (pela interface `ISerializable` ou marcando os atributos serializáveis como `SerializableAttribute`).
- **Por referência:** nessas situações, um objeto *proxy* é criado. Toda chamada de método ou manipulação de atributo é transportada do objeto *proxy* para o objeto do servidor pelo canal, para posterior resposta. A classe desses objetos deve extender `System.MarshayByRefObject`. A figura 3 mostra um exemplo conceitual das comunicações que envolvem marshaling por referência.

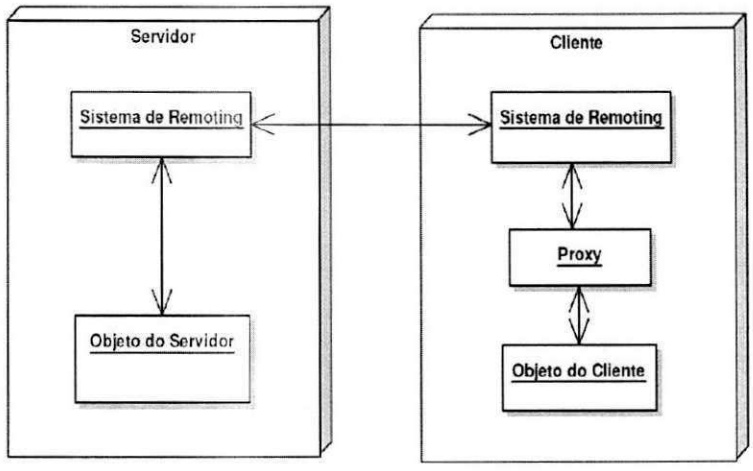

***Figura 3** - representação simplificada do marshaling por referência.*

3.2.1 Ativação dos objetos remotos

É possível ativar objetos remotos de três modos diferentes:

- **Ativado pelo cliente:** todas as instâncias de objetos remotos são criadas somente no instante em que o cliente chamar o construtor ou invocar o método `Activator.CreateInstance`. Sendo assim, haverá tantos objetos quanto a quantidade de clientes existentes. Para evitar que instâncias desnecessárias (que o cliente criou no passado, mas que ele não precisa mais manipulá-la) se acumulem na memória da máquina em que executa o servidor, é definido um "tempo de vida"(em inglês *lifetime*, convencionaremos usar duração) para cada instância criada. Mais explicações sobre o tempo de vida de um objeto remoto e como configurá-lo serão fornecidas adiante.
- **Ativado pelo servidor (*single call*):** as instâncias de objetos remotos são criadas pelo servidor, mas sua duração é somente o período necessário para a chamada de algum método, após isso ela é liberada para o garbage collector, após serem eliminadas todas as referências. A cada chamada de método, uma nova instância é criada. Por isso, não é necessário definir uma duração para esses objetos, mas também não há per se manutenção do estado dos objetos criados desse modo.
- **Ativado pelo servidor (*singleton*):** as instâncias dos objetos remotos são criadas pelo servidor, no instante da primeira chamada. Somente um objeto remoto é criado no domínio de aplicação do servidor: ainda que outros clientes façam chamadas de métodos, eles estarão utilizando a mesma instância. Esse objeto único, entretanto, pode ser liberado para o garbage collector se passar um determinado período de tempo

sem nenhuma chamada de método, segundo uma certa duração, de modo semelhante ao dos objetos remotos ativados pelo cliente.

3.2.2 Duração, leases, sponsors

Tanto no modo ativado pelo servidor via *singleton*, ou no modo ativado pelo cliente, os objetos, por *default*, não ficam ocupando o espaço de memória correspondente ao domínio de aplicação do servidor por tempo indefinido. Eles possuem uma duração que é controlada por *leases*. *Leases* (do inglês: aluguel) são períodos de tempo em que um objeto em particular esteja ativo na memória antes do sistema .NET inicia o processo de eliminação de suas referências.

Todo domínio de aplicação possui um gerenciador de leases que verifica periodicamente quais *leases* estão com o seu tempo encerrado. Caso encontre alguma nessa situação, ele notifica uma lista de *sponsors* (do inglês: patrocinador - nesse contexto *sponsor* é o cliente do objeto remoto, que decide se o *lease* deve ser renovado ou não) para aquele objeto, para ver se algum
deles renova esse lease. Se não houver nenhum que faça isso, o objeto é eliminado da memória. As configurações padrão são 5 minutos como *lease* inicial, e 2 minutos a cada chamada de método (se o atributo `RenewalOnCallTime` estiver ativo).

3.2.3 Canais e formatadores

Na infra-estrutura do .NET Remoting, tanto o cliente quanto o servidor precisam registrar um canal de comunicação, para que o *proxy* possa se comunicar com o objeto remoto no domínio de aplicação do servidor e vice-versa. Esse canal deve implementar a interface `IChannel`. A plataforma .NET já possui três tipos de canal, embora outros tipos podem ser criados por
terceiros:

- **IpcChannel** - usa *pipes* nomeados para a comunicação entre os domínios de aplicação. É mais rápido que os outros dois canais, porém tem a restrição de somente poder ser usado entre processos numa mesma máquina.
- **TcpChannel** - usa o protocolo TCP para a comunicação entre os domínios de aplicação.

- **HttpChannel** - usa o protocolo HTTP para a comunicação entre domínios de aplicação.

Existem também classes específicas para o cliente e o servidor: `IpcServerChannel`, `Ipc-ClientChannel`, `TcpServerChannel` e assim analogamente.

Ainda existem dois tipos de formatadores na biblioteca do .NET Framework que podem ser usados com Remoting:

- **`BinaryFormatter`** - serializa os atributos do objeto remoto de modo binário. É o formatador default do `TcpChannel`.
- **`SoapFormatter`** - serializa os atributos do objeto remoto como mensagens SOAP. É o formatador *default* do `HttpChannel`.

Apesar de o desempenho na maioria das vezes ser melhor com os formatadores *default*, qualquer um dos três canais permite qualquer tipo de formatador.

CAPÍTULO 4
O PROBLEMA

4. O problema

Dentro do contexto de atuação do aluno durante o estágio, foi vislumbrado um único problema a ser resolvido. Porém, dada à necessidade, houve a adição de outro problema não previsto inicialmente, por dois motivos:

- devido ao caráter de relevância e urgência deste segundo; e
- por também se utilizar a tecnologia .NET *Remoting*, a resolução de um problema pode facilitar a resolução do outro.

Antes de descrever os problemas, iremos dizer em linhas gerais os projetos aos quais os problemas estão interligados

4.1 BRSIL

O BRSIL é uma ferramenta de análise do ciclo de vida de segurança de Funções Instrumentadas de Segurança, sendo uma ferramenta de análise que utiliza modelos completos de arquiteturas redundantes durante todas as etapas de análise.

São utilizados modelos teóricos (cadeias de Markov) a ferramenta possibilita ainda o acesso a uma base de dados de equipamentos de segurança atualmente utilizada pela indústria de processos e por diversas empresas atuantes na cadeia produtiva de Petróleo e Gás Natural ao redor do mundo. A metodologia empregada no estudo das camadas de proteção está em conformidade com os padrões internacionais de aceitabilidade de risco adotados pelos demais fabricantes de sistemas dessa natureza.

O BRSIL está sendo feito através de uma parceria com a Petrobras. Ainda que seja baseado em normas mundialmente aceitas, são poucos os aplicativos encontrados que servem para função semelhante, e nenhum deles é brasileiro.

Por enquanto, o produto ainda está em desenvolvimento. Existe uma versão que executa a grande maioria das funções, a qual está sendo aperfeiçoada gradativamente. A versão atual do BRSIL está preparada para executar de forma local, mas é planejada a evolução para um ambiente distribuído.

4.2 Wireless

O objetivo do projeto Wireless consiste em investigar estratégias e soluções tecnológicas, com base em dispositivos embarcados móveis sem fio, aplicáveis à cadeia produtiva de petróleo e gás natural, e seu foco é o desenvolvimento de um sistema para aumentar o nível de controle e segurança dos procedimentos executados manualmente.

O sistema se insere na área de automação de procedimentos, monitoração e controle de fluxo de trabalho. O sistema deverá disponibilizar a integração com sistemas de supervisão e controle de processos, com interfaces padronizadas de modo que as aplicações clientes possam interagir com um aplicativo servidor para acesso a recursos do sistema supervisório corporativo.

Este sistema toma como base a comunicação entre dispositivos móveis (PDAs, Tablets) com servidores *web* para a realização de tarefas comuns nas plantas de refino, assim como a sincronização de dados em ambos os sentidos (da central para o operador em campo, e vice-versa).

O projeto Wireless está em sua fase final.

4.3 Problema principal

Atualmente o BRSIL, apesar de no presente momento estar voltado para aplicações locais, o sistema está dividido em duas porções, que se comunicam entre si através de um ASP .NET *web service*, na estrutura cliente-servidor.

Há uma necessidade de se avaliar outras formas de comunicação entre processos, tendo em vista as seguintes possibilidades:

- funcionamento de modo distribuído;
- aumento da flexibilidade na troca de porções do sistema;
- aumento da eficiência do sistema em virtude da diminuição do tempo de resposta; e
- criação de interface para comunicação com outros sistemas futuros, ou já existentes (através de adaptações).

Pelos motivos citados acima, o estudo do comportamento de diversos *frameworks* para a comunicação entre processos foram estudados, entre eles o de .NET Remoting.

O problema a ser resolvido, no contexto deste estágio, é estudar a tecnologia de comunicação .NET Remoting e analisar o seu uso no BRSIL, tanto em termos de viabilidade quanto de utilidade para o aperfeiçoamento do sistema como modelo de análise.

4.4 Problema secundário

O projeto Wireless também utiliza em seu funcionamento Web Services. Esses *web services* freqüentemente precisam se utilizar de dados que não podem ser tão volúveis quanto o tempo de duração da sessão de um *web service*, porém não há a necessidade de serem gravados em disco, por questão de eficiência (quando dados necessitam de persistência, eles são comumente armazenados em banco de dados, no caso do Wireless).

Também há a necessidade de intercomunicação com outros componentes executáveis de forma independente, como gerenciadores de configuração de registro (*log*) de eventos relevantes. Para essas situações, foi vista como solução útil também a inclusão de .NET Remoting.

CAPÍTULO 5

ESTUDO DO .NET REMOTING

5. Estudo do .NET Remoting

A primeira fase do estágio feita foi o estudo da tecnologia .NET Remoting. Exceto pelas áreas mais complexas, o estudo foi acompanhado de desenvolvimento de aplicações simples para o melhor entendimento do assunto.

Foram analisados os tipos diferentes de canais, se o modo de transmissão era feito através de IPC, TCP ou HTTP. Foi medido o tempo de resposta de um mesmo método através de objetos remotos compartilhados através de cada um desses tipos de canais. O resultado obtido, assim como esperado, confirma que o canal ICP (através de *pipes* do sistema operacional), é mais veloz que o TCP. .NET Remoting através de HTTP foi o protocolo de comunicação com maior tempo de resposta, conforme pode ser visto na figura 4 e na tabela 2.

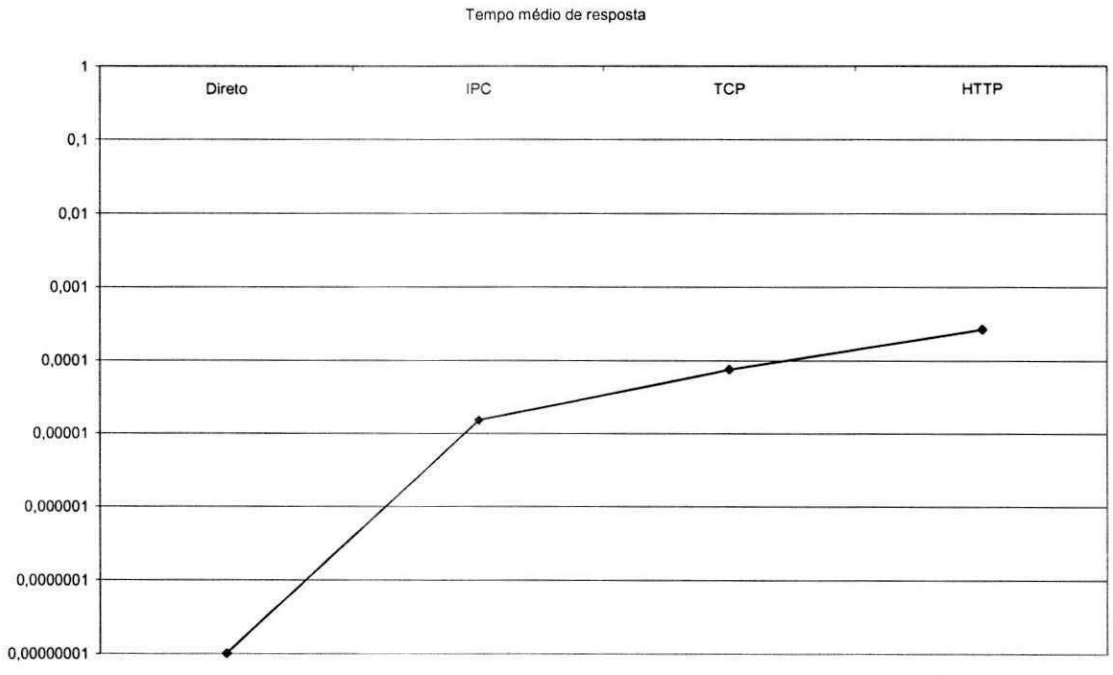

Figura 4 - *valores médios de tempo de resposta através de cada canal.(gráfico em escala logarítmica*

	Tempo médio (em segundos)	Número de tentativas sucessivas
Acesso direto (objeto no mesmo domínio de aplicação)	$1{,}003 \cdot 10^{-8}$	10^9
IPC (*binary formatter*)	$1{,}502 \cdot 10^{-5}$	10^6
TCP (*binary formatter*)	$7{,}515 \cdot 10^{-5}$	10^5
HTTP (*SOAP formatter*)	$2{,}672 \cdot 10^{-4}$	10^5

Tabela 2: *tempo médio de duração da chamada de método simples, assim como o número de tentativas realizadas, para cada tipo de canal no .NET Remoting. Percebe-se que, apesar de o IPC ser o meio mais rápido, ainda é da ordem de 103 mais lento que executado em um único domínio de aplicação.*

Como dito na seção dos conceitos, na hora de estabelecer a comunicação entre os diversos domínios de aplicação, podem ser usados três tipos de ativação: ativado pelo cliente, ativado pelo servidor no modo *single call*, e ativado pelo servidor no modo *singleton*. Dentre essas três, segundo LÖWY (2003), o que é menos oneroso para a máquina que contém o servidor é o modo ativado pelo cliente, porque a área de memória reservada para o objeto criado através desse modo pode ser liberada logo após a execução da chamada de método, enquanto objetos não utilizados podem se acumular se forem ativados pelo cliente, e em menor grau no modo *singleton*. O modo *single call* também tem a vantagem de permitir mais facilmente a replicação de servidores.

Os dados relativos aos atributos e métodos de um objeto que pode ser disponibilizado remotamente podem ser extraídos de uma ferramenta chamada soapsuds, ou pode ser usado o padrão de projeto *Factory Method* para criar instâncias de um objeto com uma interface pré-definida a partir de um método de outro objeto (*Factory*). Caso seja optada pela opção de usar o soapsuds, os dados do canal e do objeto remoto devem ser armazenados num arquivo que segue a nomenclatura <nome da aplicação>.config.

Objetos remotos podem ser disponibilizados através da forma que o programador desejar, por exemplo, através de programas executáveis, ou serviços; ou ainda, pode dispor esses objetos através do IIS. Quando a possibilidade de que esses objetos remotos possam ser utilizados por diversas outras máquinas em uma rede, é provável que a melhor escolha seja utilizando o IIS, visto que assim o programador não

precisa se preocupar com serviços como autenticação ou conexão segura. Entretanto, quando o IIS é utilizado, apenas o canal do tipo HTTP pode ser utilizado. Para diminuir o tempo de resposta, pode ser utilizado a formação binária, ao invés do uso de SOAP.

.NET Remoting permite o uso de conexões seguras, em especial se utilizado no IIS, inclusive com o protocolo SSL. Entretanto, nada impede que o programador possa construir seu próprio mecanismo de autenticação e confidencialidade nas chamadas a objetos remotos. .NET remoting permite a pessoa construir aplicações através de protocolos de segurança como NTLM, Kerberos ou SPNEGO.

CAPÍTULO 6
IMPLEMENTAÇÃO DO .NET REMOTING NO WIRELESS

6. Implementação do .NET Remoting no Wireless

6.1 Versão Wireless sem remoting

De forma simplificada, o sistema Wireless continha as seguintes porções conceituais, conforme pode ser visto na figura 5: um **módulo do cliente móvel**, onde um operador recebia informações sobre as operações que ele deveria fazer, e ao mesmo tempo, serve para acompanhar o processo de realização das tarefas a ele repassadas, e ao fim de sua execução, informar ao servidor de dados os resultados correspondentes. Um **módulo de aplicação web**, que serve de elo entre o módulo do cliente móvel e o módulo de dados. O módulo de aplicação web fica localizado em um computador em uma zona "desmilitarizada" da rede, para dificultar o acesso indevido a partes mais importantes da rede. O módulo de aplicação web, por si, está ligado ao **módulo de dados**, sendo este o qual mantém a maior parte dos dados relevantes para o processo de inspeção de equipamentos na planta da indústria. Estão ligados a este os seguintes módulos: o módulo de **cadastro e acompanhamento**, utilizado pela gerência a fim de cadastrar as ordens de serviço (atividades comuns que devem realizadas pelo operador), assim como observar a evolução do cumprimento das ordens de serviço enquanto elas estão sendo realizadas. O módulo de automação serve como intermediário para se comunicar com a parte de automação da indústria, ou seja, com os dispositivos que fazem medições e avaliações do comportamento da indústria de forma automática, como sensores e controladores eletrônicos. Para efeitos deste relatório de estágio, destacaremos apenas os módulos de cliente móvel, de aplicação web e de dados, pois foram nestes módulos em que foi dedicada a maior parte do trabalho do aluno.

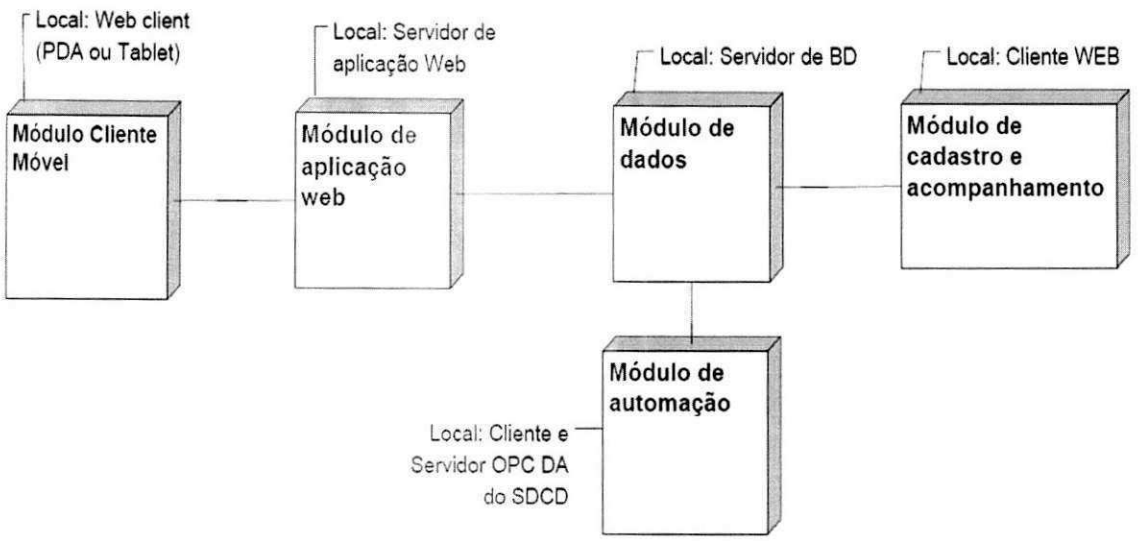

Figura 5: *Descrição simplificada do sistema Wireless.*

Todos os módulos foram concebidos para que pudessem ser substituídos de forma que não comprometessem o modo de execução nas camadas vizinhas, como por exemplo, através da mudança de assinatura de interfaces ou novas compilações.

Por esse motivo, a arquitetura adotada se aproxima das definições mais comuns de SOA (arquitetura orientada a serviços, do inglês *service oriented architecture),* no qual a interface não deve ser muito acoplada a um contrato exclusivo entre duas entidades do mesmo tipo. Isso foi feito através da adoção de um conjunto restrito de métodos, que são:

- `GetChavePublica`: serve para que a entidade cliente possa obter a chave criptográfica do tipo assimétrica pública do seu respectivo servidor, e assim possa seguir para o uso do método `Subscribe`.
- `Subscribe`: nesse método, o cliente fornece ao servidor sua identidade para que as duas entidades possam trocar chaves criptográficas do tipo simétrica, para uso durante uma sessão.
- `Read`: método utilizado para que a entidade cliente possa obter informações do servidor sobre algo específico, através da adição de parâmetros de tipo e de filtro.
- `Write`: semelhante ao método `read`, porém junto aos parâmetros anteriores, também é fornecido algum dado de entrada que provavelmente servirá para ser gravado no servidor. O retorno do método é uma mensagem de confirmação de que o método foi realizado de forma esperada, além do resultado de uma possível consulta anexa à gravação.

A interface de ambos os servidores é a mesma, isto implica dizer que o servidor de aplicação se comporta como servidor em relação ao cliente PDA, porém como cliente em relação ao servidor de dados. Para que os serviços de `Read` e `Write` possam ser amplamente versáteis, além de diminuir a dependência de uma plataforma ou tecnologia específica, os parâmetros fornecidos são simplesmente o identificador de sessão (um número inteiro), assim como uma *string* codificada no formato XML criptografada. O retorno dos métodos também é uma string codificada em XML e depois criptografada. Tanto a sintaxe do XML enviado quanto do recebido possui uma parte fixa e outra flexível, que varia de acordo com cada caso de uso existente, sendo possível até a inclusão de novos casos de uso e adaptação de parâmetros sem necessidade de criar mais serviços. Os clientes podem ter acesso à descrição dessa sintaxe flexível através de arquivos XSD.

Inicialmente, tanto o servidor de aplicação quanto o banco de dados não utilizavam .NET Remoting. Toda a lógica de negócio era realizada através de *web services*. Apesar de o sistema funcionar relativamente bem, algumas vezes ocorriam problemas com a manutenção de sessão, por exemplo, das chaves criptográficas simétricas perderem a validade antes do momento desejável, ou quando o IIS era reinicializado. Também se planejava incluir outras porções ao sistema, como por exemplo um gerenciador de configuração. Por razões de segurança e de separação de domínios, essas partes adicionais não poderiam se comunicar através do IIS, mas sim de modo independente. Todas essas necessidades poderiam ser solucionadas através da separação da lógica de negócio do *web service*, e isso foi feito através de .NET Remoting.

6.2 Inclusão de .NET Remoting no Wireless

O primeiro passo foi separar o *web service* da lógica de negócio. Foi necessária então uma refatoração do código. Todo o código da lógica de negócio foi posto em uma biblioteca (DLL). Foi utilizado o padrão de projeto *Façade*, ou seja, todas os atributos e métodos que cuja intenção é de serem visíveis externamente estão localizados em uma mesma classe, a "fachada" ou *façade*. Essa fachada repassa então os parâmetros para os objetos de outras classes, que executam as devidas operações, sem a necessidade de que esses outros objetos sejam visíveis para o ambiente externo ao da biblioteca. Essa medida também facilitou o uso de .NET Remoting, pois assim apenas um objeto precisa

ser compartilhado remotamente através do canal. Para o objeto da fachada, foi adotado como duração (*lifetime*) para o *leasing* de 24 horas, que é o intervalo máximo entre dois turnos de um mesmo operador.

Em seguida, foi necessário criar o servidor do canal de Remoting. Para que o sistema pudesse funcionar com maior velocidade e, dado que o IIS e qualquer outro aplicativo que também viesse a utilizar o sistema deve executar apenas em uma mesma máquina, foi utilizado o canal do tipo IPC (`IPCServerChannel`), configurado de modo programático (através de código, e não por meio de arquivos de configuração). O objeto remoto, no caso a fachada da DLL, foi configurado para ser ativado pelo servidor no modo Singleton, pois se há um único repositório de chaves de sessão, não há a necessidade de manter várias cópias da mesma informação em objetos diferentes (caso se tivesse optado por objetos ativados pelo cliente), nem essas chaves são duradouras o suficiente para exigir gravação em meio persistente (caso os objetos fossem ativados pelo servidor em modo single call).

Entretanto, existe um problema adicional. Os ASP .NET web services normalmente executam suas aplicações como um usuário específico para esse fim, comumente nomeado ASPNET. Entretanto, por *default*, o usuário ASPNET não possui permissão de usar o canal do tipo IPC. Entretanto, é possível contornar esse problema pela modificação das configurações do canal:

- Criou-se uma tabela *Hash* de configurações.
- Nessa tabela *Hash*, foram postos os dados do canal, como o seu nome e a porta onde ele está ligado.
- Adicionalmente, foi necessário adicionar uma propriedade chamada de "authorizedGroup", que se refere ao grupo de usuários do Windows que pode executar aplicações através daquele canal. Para essa propriedade, chamamos de um nome qualquer, neste exemplo "remoting users".

```
Hashtable properties = new Hashtable();
properties["name"] = "nome_do_servidor";
properties["authorizedGroup"] = "remoting users";
properties["portName"] = "nome_da_porta";
```

- Em seguida, foi necessário criar um grupo de usuários no Windows com o mesmo nome (remoting users), e nesse grupo incluir todos os usuários humanos autorizados a manipular o sistema Wireless, assim como o usuário ASPNET.
- Para que o *web service* possa ter uma referência para o objeto remoto, é necessário criar um arquivo de aplicação ASP .NET (normalmente de nome

Global.asax), e nele incluir no método Application_Start a criação do canal cliente (do tipo `IPCClientChannel`), criar uma instância do objeto remoto e armazená-la na aplicação. Desse modo, mesmo interrompendo-se o IIS por algum momento, o estado é mantido no servidor do canal de remoting.

Os mesmos passos foram seguidos tanto para o servidor de banco de dados quanto o servidor de aplicação web.

6.3 Funcionamento do sistema Wireless com .NET Remoting

Esta subseção mostra como o sistema está organizado após a adição de .NET Remoting. O servidor do canal de .NET Remoting é um aplicativo executável comum, que deve estar em constante funcionamento para que haja comunicação efetiva entre as entidades do sistema. A figura 6 mostra através de um diagrama de atividade o funcionamento do servidores do canal IPC do .NET Remoting.

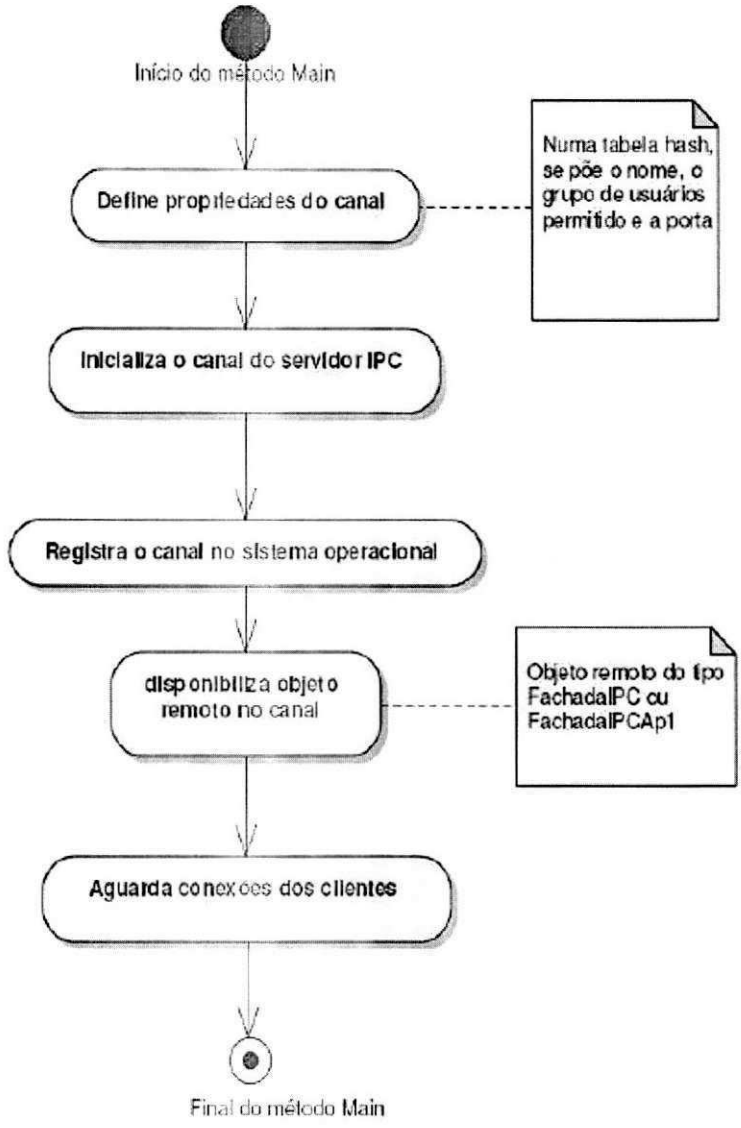

***Figura 6** - diagrama de atividade do servidor de .NET Remoting*

Nas figuras 7 e 8 a seguir pode ser visto como ficou a arquitetura interna do módulo de aplicação web e do módulo de dados, assim como o comportamento em operações `Read` e `Write`. Pode-se perceber que os web services agora não mais possuem lógica de negócio, apenas servido como intermediários para o funcionamento do sistema.

Outra fato perceptível é que há um outro componente de software ligado ao servidor de .NET Remoting de modo independente ao *web service*, que são as aplicações de gerência, tanto no módulo de aplicação *web* quanto de banco de dados.

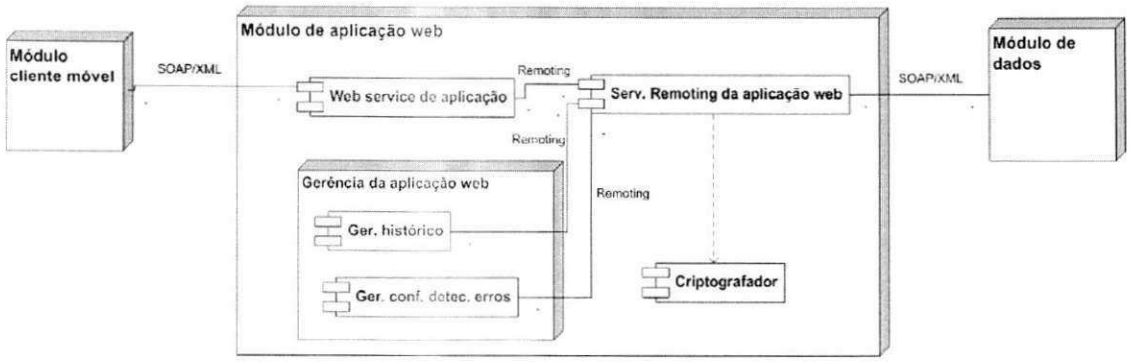

***Figura** 7 - arquitetura do módulo de aplicação web.*

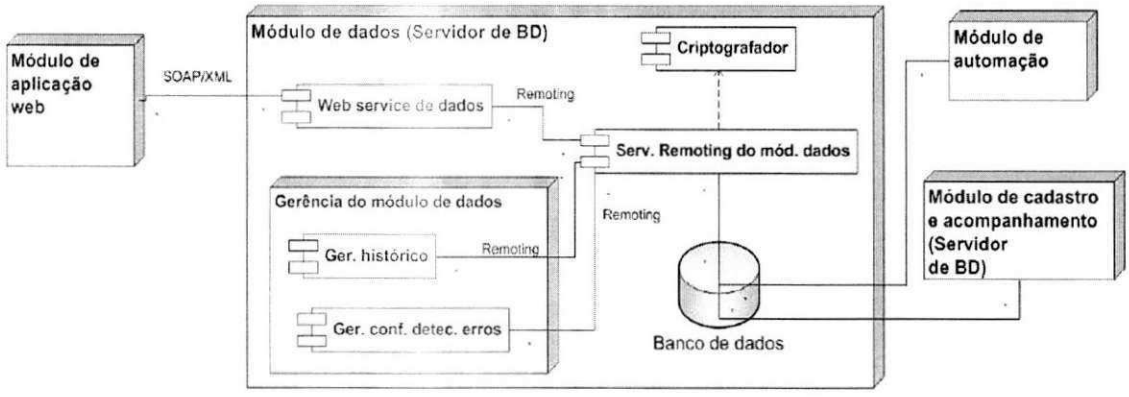

***Figura** 8 - arquitetura do módulo de dados.*

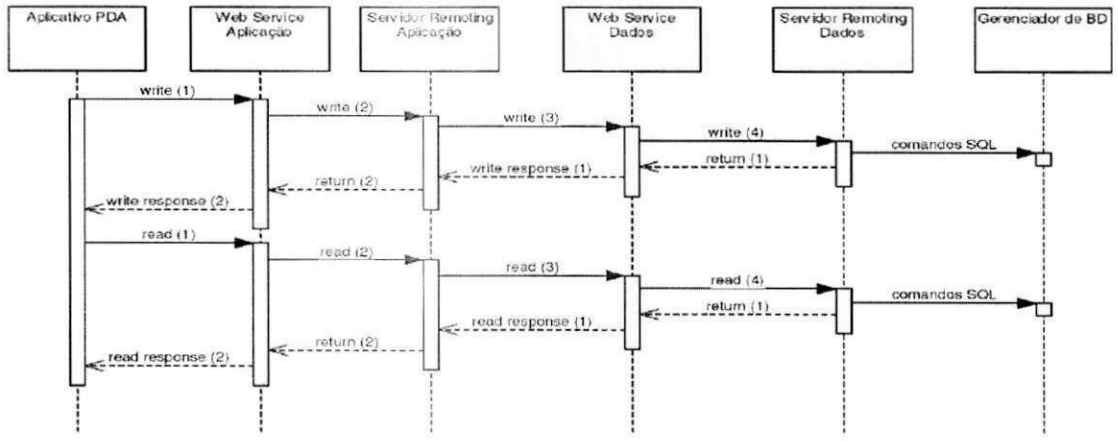

***Figura 9** - diagrama de seqüência para os serviços de read e write.*

Portanto, podemos perceber por meio deste sistema como .NET Remoting pode resolver diversos problemas que envolvam intercomunicação entre processos.

6.4 Trabalhos adicionais no Wireless

O aluno também trabalhou no projeto Wireless na concepção de alguns caso de uso adicionais, como na inclusão do mecanismo de *polling* (verificação periódica de novas informações) através do método read, assim como na implementação de um banco de dados para o PDA, servindo como armazenamento temporário de informações em situações que o PDA perde o sinal da rede. Além disso, corrigiu eventuais *bugs* detectados na ferramenta.

CAPÍTULO 7

Avaliação do .NET Remoting no BRSIL

7. Avaliação do .NET Remoting no BRSIL

Levando-se em consideração o tempo gasto no projeto Wireless, e de acordo com o plano de estágio, o uso de .NET Remoting não foi diretamente implementado no BRSIL até o presente momento.

A arquitetura do BRSIL, dada às suas funcionalidades não necessitarem ainda de uma maior distribuição física, é relativamente simples, um caso particular da arquitetura cliente-servidor.

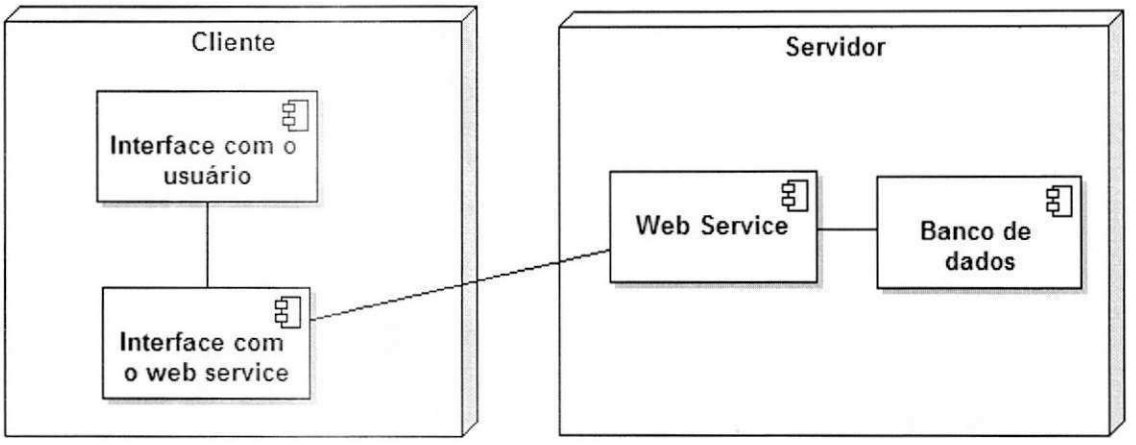

***Figura** 10 - diagrama da arquitetura do BRSIL*

A parte da interface com o usuário está bem delimitada e separada da parte que se comunica com o *web service* (semelhante à separação do *view* do *controller* no padrão arquitetural MVC). Apenas o *Web Service* tem acesso direto ao Banco de dados. Porém, é justamente no web service onde está o ponto fraco do desempenho do sistema. Como são comumente os ASP .NET Web Services, a transmissão de dados é feita através do protocolo SOAP sobre HTTP. Isso traz vantagens em sistemas voltados para uma rede, mas não para aplicações em modo local. Como pôde ser visto anteriormente, o protocolo HTTP junto com SOAP gera um tempo de resposta maior do que em outras alternativas. Porém, como o modo local é um degrau para o funcionamento através de uma rede corporativa, a escolha é compreensível.

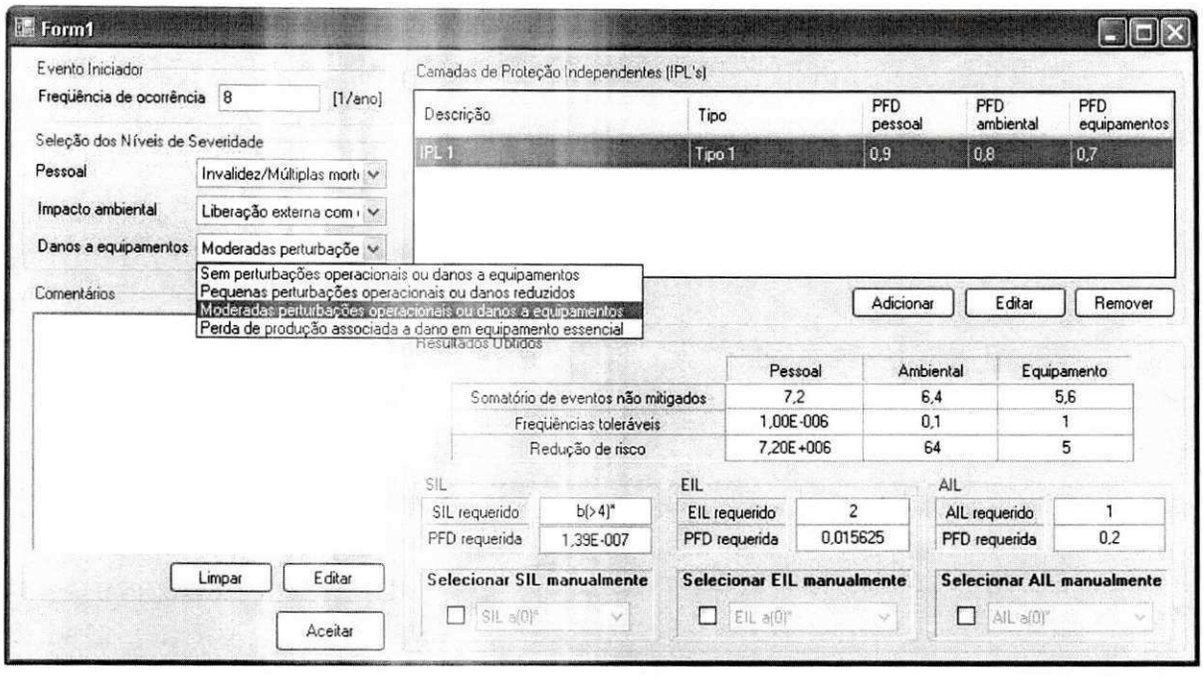

Figura 11 - exemplo de tela do BRSIL.

Portanto, em termos de desempenho, não há a necessidade de se acrescentar um nível a mais de comunicação por meio de .NET Remoting, caso o BRSIL se mantenha estritamente uma ferramenta de análise de risco com entrada manual de dados. Entretanto, não é possível descartar a possibilidade que a ferramenta se integre com outros processos, como por exemplo, com o próprio sistema instrumentado de segurança, por meio de uma realimentação automática. Então, caso se decida por usar .NET Remoting, são recomendadas as seguintes medidas:

- Caso não haja necessidade de integração do BRSIL, de preferência o acesso ao banco de dados deve ser feito de forma direta (provavelmente através de uma DLL específica para tal).
- Caso a intercomunicação entre processos seja importante, que se dê a preferência por canais com transporte IPC, se for na mesma máquina.
- Se o .NET Remoting for utilizado em uma rede local ou através da internet, a melhor alternativa é abrigando o servidor no IIS, portanto usando canais HTTP com formatação binária.
- Na implementação atual, não há manutenção de estado, então o melhor modo de ativação para esse fim é ativado pelo servidor *single call*.

- Se não for um requisito do cliente manter a interoperabilidade os *web services* podem ser eliminados.
- Outro fato relevante é que, para substituir a lógica do web service pela lógica contida num processo distinto, é necessária a criação de uma DLL com uma interface comum, provavelmente seguindo o padrão *façade* (como foi no Wireless).
- Caso haja uma demanda relativamente alta de pedidos a um único servidor, causando tempos de resposta elevados em períodos de maior tráfego, é sugerido considerar o uso de mais de um servidor. .NET Remoting permite o balanceamento de carga. Objetos remotos da mesma classe podem ser criados em servidores distintos que tratam uma finalidade comum.

CAPÍTULO 8
CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

8. Considerações finais e trabalhos futuros

Este estágio foi muito importante para o aprendizado do aluno em seu ambiente de trabalho, tanto em virtude da pesquisa sobre .NET Remoting, assim como em outras tecnologias, como SOA, Bancos de dados, mensuração, dispositivos móveis etc.

Do ponto de vista dos projetos envolvidos, o Wireless foi onde se pôde ver mais resultados. A aplicação está totalmente funcional, necessitando apenas de alguns incrementos mínimos. No caso do BRSIL, como a ferramenta ainda estava em desenvolvimento no período do estágio do aluno, não foi prioritária a alteração do código, mas as recomendações certamente serão úteis para nortear andamento do projeto no futuro.

Os supervisores cumpriram com seu papel, de tirar dúvidas sobre quais medidas serem tomadas nas decisões mais complexas durante o semestre, propiciando o cumprimento das atividades com menor risco de erros de planejamento.

Como trabalhos futuros, existem diversos rumos possíveis. Um deles seria o de tentar implementar as diversas formas de canais sugeridos e testar o seu desempenho na prática. Outra possibilidade é a de aproveitar os conhecimentos adquiridos em .NET Remoting para aplicar no estudo da tecnologia WCF (*Windows Communication Foundation)*, a mais nova tecnologia da plataforma .NET para comunicação entre processos, englobando as finalidades de outras tecnologias como web services e .NET Remoting e facilitando o desenvolvimento de aplicações com arquitetura orientada a serviço.

Bibliografia

LÖWY, D. ***Programming .NET Components.*** 1. ed. Sebastopol, O'Reilly, 2003.

RAMMER, I. ***From .NET Remoting to the Windows Communication Foundation (WCF)***. Disponível em: http://msdn2.microsoft.com/en-us/library/aa730857(vs.80).aspx. Último acesso: 26 setembro 2007.

_____ & SZPUSZTA, M. ***Advanced .NET Remoting.*** 2. ed. New York, Apress, 2005.