UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

"Uma Arquitetura Orientada a Serviços para Redes de Monitoração e Atuação Sem Fio Utilizando SOA"

Jeferson Martin de Araujo

São Carlos
Setembro/2008

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

"Uma Arquitetura Orientada a Serviços para Redes de Monitoração e Atuação Sem Fio Utilizando SOA"

Jeferson Martin de Araujo

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação, da Universidade Federal de São Carlos, como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação.
Área de concentração: Sistemas Distribuídos e Redes.
Orientador: Prof. Dr. Luis Carlos Trevelin

São Carlos
Setembro/2008

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

A663ua

Araujo, Jeferson Martin de.
Uma arquitetura orientada a serviços para redes de monitoração e atuação sem fio utilizando SOA / Jeferson Martin de Araujo. -- São Carlos : UFSCar, 2009.
130 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2008.

1. Arquitetura de redes de computadores. 2. Arquiteturas distribuídas. 3. Arquitetura - inovações tecnológicas. 4. Arquitetura de software. I. Título.

CDD: 004.68 (20ª)

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

"Uma Arquitetura Orientada a Serviços para Redes de Monitoração e Atuação sem Fio utilizando SOA"

JEFERSON MARTIN DE ARAUJO

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação

Membros da Banca:

Prof. Dr. Luis Carlos Trevelin
(Orientador - DC/UFSCar)

Profa. Dra. Regina Borges de Araújo
(DC/UFSCar)

Profa. Dra. Linnyer Beatrys Ruiz
(UEM)

São Carlos
Setembro/2008

DEDICATÓRIA

Aos meus pais, Wanderley e Roseli, pelo apoio e incentivo do começo ao fim deste trabalho.

AGRADECIMENTOS

Percorre-se um longo caminho até chegar ao final de um Mestrado. Muitas pessoas cruzam nossas vidas ao longo desse caminho, algumas apenas passando ao largo, outras mudando rumos. Muitas dificultam o caminho, outras o tornam possível. É impossível, lembrar de todos os que, direta ou indiretamente, deixaram uma marca nesse longo trajeto. Porém, tentarei deixar aqui registrados meus agradecimentos àqueles que me acompanharam, de alguma forma ajudaram, e tornaram possível esse momento tão desejado.

Ao meu orientador Luis Carlos Trevelin, pela oportunidade concedida, orientação e confiança.

À minha família, agradeço imensamente, pelo apoio incondicional.

Aos amigos da BM&F, em primeiro lugar, por uma questão de cronologia, gostaria de agradecer ao Mariano e ao Eirado. Em seguida, ao Sergio Yamani, meu chefe por quase 3 anos e que me permitiu conciliar o duro trabalho na BM&F com as atividades do Mestrado. Aos amigos Luiz Fernando (Ryche) e Jefferson Marchetti, pelas idéias e dificuldades compartilhadas. E finalmente, também por uma questão cronológica, porque termino um caminho e começo outros, gostaria de agradecer ao Marcio Castro pelo entusiasmo que sempre me contagiou.

À Patrícia, minha namorada, pela paciência e carinho.

RESUMO

A pesquisa em redes de sensores é considerada uma das áreas que mais apresenta desafios e com maior potencial para crescimento. Ao combinar a teoria e conceitos da Ciência da Computação com a aplicabilidade da Engenharia Elétrica, as Redes de Sensores e Atuadores Sem Fio (RSASF) apresentam uma enorme variedade de estudos de casos para o desenvolvimento de aplicações que auxiliam na evolução da arquitetura, infraestrutura e dos componentes em geral dessas redes.

Devido aos avanços tecnológicos resultantes dessas pesquisas cientificas, as unidades de processamento tornaram-se exponencialmente mais poderosas, de menor tamanho físico e com menor custo financeiro, impulsionando a utilização das RSASF e tornando-as cada vez mais populares. Desse modo, é evidente o surgimento de aplicações que utilizam ao menos algum dos conceitos e características inerentes às RSASF como infraestrutura para o desenvolvimento de sistemas nos mais variados domínios de aplicação. Atualmente, existem aplicações dos conceitos das RSASF desde monitoramento ambiental até utilização em campo militar.

Um dos maiores problemas associados à RSASF está na dificuldade do desenvolvimento de aplicações para estas redes, derivados do fato da existência de grande heterogeneidade de dispositivos e a falta de padronização de codificação. Pesquisas recentes têm demonstrado que modelos de programação orientados a serviços podem contribuir para melhorar a comunicação e interoperabilidade entre esses dispositivos heterogêneos. O desenvolvimento de aplicações para estas redes provê vantagens competitivas com o uso de Arquiteturas Orientadas a Serviços (SOA – *Service-oriented Architecture*) ao desenvolver aplicações em uma arquitetura que suporta reusabilidade, manutenção facilitada, gerenciamento e estensibilidade. Dessa forma, este trabalho propõe o uso de uma abordagem orientada a serviços para o desenvolvimento e gerenciamento de aplicações com fins de atuação e monitoramento que utilizam conceitos e características das RSASF.

Palavras-chave: Redes de Sensores e Atuadores Sem Fio, Arquitetura Orientada a Serviços, Vigilância e Monitoramento Remotos, Arquitetura de Soluções e Aplicações, Arcabouços de desenvolvimento

ABSTRACT

Research in sensor networks is widely regarded both as one of the most challenging and prospering area. Matching Computer Science concepts with Electrical Engineer applicability, Wireless Sensor and Actors Networks (WSAN) can be used to develop several case studies in manner to build applications which will be useful for evolving the WSAN's architecture, infra-structure and its components in general.

Due to technologies advances reached by these scientific researches, the processing unities became exponentially more powerful with smaller size and lower cost, enabling the widespread use of the WSAN and making this technology more popular. So, it is clear that applications which depend on the WSAN's infrastructure to build system, in several applications domains, have been emerging. Actually, there are applications ranging from habitat monitoring to military purposes.

One of the major problems associated with WSAN is the difficulty of developing software applications. Recent researches has shown that service oriented programming models may contribute to improve communication and interoperability between heterogeneous devices in pervasive spaces. Organizations seeking to develop WSAN applications can obtain a competitive advantage by developing applications on an architecture which supports reusability, maintainability and extendibility. Therefore, this work proposes the use of a service oriented approach for the development and management of applications specialized in remote area surveillance based on wireless and sensor networks characteristics and concepts.

Keyworks: Wireless Sensor and Actor Networks, Service Oriented Architecture, Remote Surveillance and Monitoring, Application and Solution Architectures, Frameworks.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

CAPÍTULO 1 – INTRODUÇÃO

Neste capítulo, apresenta-se o escopo, objetivo, a motivação e relevância e a estrutura do trabalho da presente tese. A seção 1.1 contextualiza este trabalho. As seções 1.2 e 1.3 introduzem os principais objetivos e a motivação e relevância da tese. Finalmente, a seção 1.4 contém a estrutura deste manuscrito.

1.1 Apresentação

O padrão IEEE 1471-2000 [1] define arquitetura como a organização fundamental de um sistema através de seus componentes, as relações entre eles e com o ambiente, e os princípios para guiar o projeto e a evolução de um sistema ao longo do tempo. Neste sentido, um sistema é uma coleção de componentes organizados a fim de atender um conjunto de funções específicas.

Modelos de referência, padrões arquiteturais e arquiteturas de referência representam decisões de projeto iniciais e são aplicados para capturar componentes arquiteturais, padrões arquiteturais e estilos arquiteturais [2].

Este trabalho apresenta o desenvolvimento de uma arquitetura baseada no modelo de referência OASIS para SOA [3]. São detalhados os passos para a aplicação desta metodologia de acordo com as definições básicas de [4] e, para cada um destes passos, é apresentado o resultado correspondente, sendo que ao final é apresentada uma proposta de arquitetura para hospedar uma rede sem fio, baseada em conceitos e características das Redes de Sensores e Atuadores sem Fio (RSASF), com fins de atuação e monitoração.

Sistemas baseados em Redes de Sensores e Atuadores sem Fio poderiam beneficiar-se do uso de uma arquitetura orientada a serviços (SOA – *Service-oriented Architecture*) que fornecesse um ambiente de execução genérico para as aplicações, provendo uma abstração das funcionalidades de infraestrutura dessas redes. Entre outras funções, os

mecanismos e serviços provenientes dessa arquitetura poderiam decidir o melhor protocolo a ser usado de acordo com os requisitos da aplicação, coordenar a operação dos sensores na realização do objetivo dessa aplicação, gerenciar o uso dos recursos da rede, controlar a segurança da rede, etc. Além disso, a fim de prover de forma eficiente a qualidade requerida pelas aplicações, muitas vezes são necessários serviços para interagir com os níveis mais baixos da pilha de protocolos dos sensores, ou mesmo com dispositivos de hardware.

As Redes de Sensores e Atuadores Sem Fio possuem características particulares que as diferem de redes de computadores tradicionais em vários aspectos, apresentando, assim, desafios únicos. Primeiramente, de forma a tornar as redes de sensores economicamente viáveis, os mecanismos sensores são limitados em suas capacidades de consumo de energia, de computação, comunicação e de armazenamento de dados. Segundo, ao contrário de redes tradicionais, os nodos sensores podem estar distribuídos em áreas acessíveis, apresentando o risco adicional de ataque físico [5].

Com a miniaturização dos componentes eletrônicos e dos componentes de hardware, paralelamente à evolução da comunicação sem fio, o desenvolvimento e o uso das Redes de Sensores e Atuadores sem Fio vêm crescendo, já tendo aplicações práticas em diferentes áreas, tais como biológicas e médicas [6, 7], ambientais [8], agricultura, pecuária [9], mineração [10], climáticas, químicas, militares, segurança [11], dentre outras [12]. Assim, as RSASF têm sido viabilizadas pela rápida convergência e pelos avanços de quatro tecnologias: microprocessadores, micro-controladores, comunicação sem fio e micro sistemas eletromecânicos [13].

As RSASF são compostas, basicamente, por vários nodos de sensoriamento (os nodos sensores, também chamados apenas de sensores), pelos nodos atores ou atuadores, que são capazes de atuar no ambiente, por um ou mais pontos de concentração de informação (esses nodos especiais são chamados de sorvedouros), que possuem maior poder de processamento e por um nodo *gateway*, cujo objetivo é fazer a interface dos dados recebidos pelo(s) nodo(s) sorvedouro(s) com uma aplicação cliente final. Pensando isoladamente, o trabalho de um nodo sensor é basicamente captar informações (através dos seus dispositivos sensores), processar essas informações e transmití-las [14]. Já os nodos sorvedouros podem ter comunicação com o mundo externo (Internet, por exemplo) através do nodo *gateway*, para que o usuário possa monitorar a rede de sensores remotamente. A figura 1 ilustra esta configuração de RSASF.

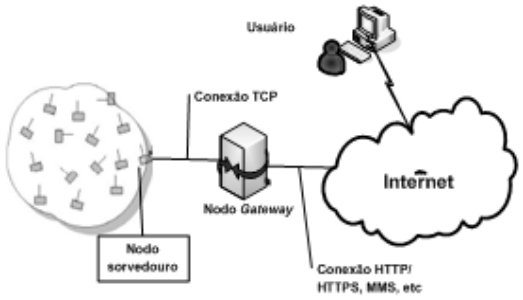

Figura 1: Representação de uma possível configuração de uma RSASF

Do ponto de vista científico, as RSASF apresentam uma grande variedade de novos problemas não estudados ou ainda incipientes. No relatório do Workshop sobre pesquisas fundamentais na área de redes, patrocinado pela *National Science Foundation* (NSF) [15], a pesquisa em redes de sensores foi considerada uma das seis áreas de grande desafio de pesquisa. Outra área relacionada foi a de teoria de redes de comunicação sem fio, que engloba as RSASF. Redes de Sensores e Atuadores Sem Fio são um dos primeiros exemplos no mundo real de computação "pervasiva", i.e., a noção de que dispositivos de computação e sensoriamento pequenos, inteligentes e baratos irão eventualmente permear o ambiente. Esta noção tem infiltrado os círculos da Tecnologia da Informação por mais de uma década [16]. Dentre esses problemas, podemos elencar a dificuldade no desenvolvimento de aplicações para estas redes, derivados do fato da existência de grande heterogeneidade de dispositivos e a falta de padronização de codificação. Além disso, uma RSASF é um tipo de sistema dependente da aplicação, ou seja, os parâmetros de configuração, operação, gerenciamento e manutenção variam com os objetivos da aplicação. No entanto, as Redes de Sensores e Atuadores Sem Fio devem ser flexíveis o suficiente para se adaptarem a variados tipos de aplicações, tais como atuação e monitoramento de residências, hospitais, olarias, indústrias, etc. sem grandes alterações em sua arquitetura e infraestrutura. Em outras palavras, a complexidade do ambiente deve ser abstraída.

Pesquisas recentes têm demonstrado que modelos de programação orientados a serviços podem contribuir para melhorar a comunicação e interoperabilidade entre dispositivos heterogêneos, como os utilizados em uma RSASF. O desenvolvimento de aplicações para estas redes provê vantagens competitivas com o uso de SOA ao desenvolver

aplicações em uma arquitetura que suporta reusabilidade, manutenção facilitada, gerenciamento e estensibilidade.

Uma RSASF hospedada em uma arquitetura orientada a serviços possui como benefícios principais:

a) gerenciamento dinâmico, independente da aplicação;

b) a inclusão de novos nodos e novas funcionalidades e a facilidade de desenvolver novas aplicações para extrair as informações necessárias da rede e usá-las no contexto do sistema que está sendo desenvolvido;

c) utilização e reuso de componentes genéricos ou de infraestrutura.

Neste sentido, este trabalho almeja a proposição de uma arquitetura orientada a serviços para a construção de um sistema hábil a automatizar e monitorar eventos remotamente, utilizando os conceitos de Redes de Sensores e Atuadores Sem Fio, e que seja modular, no sentido de que essa arquitetura proposta seja facilmente adaptável a qualquer ambiente ou cenário da aplicação proposta.

Para melhor assimilar os conceitos e justificar os resultados, nesse trabalho também propusemos a construção de um estudo de caso, o qual permitirá também aplicar na prática estudos teóricos e ainda alavancar a pesquisa na busca por uma arquitetura e infraestrutura que sejam capazes de atender aos pré-requisitos técnicos e financeiros que irão nortear esse estudo.

1.2 Objetivos

O objetivo central deste trabalho é propor uma arquitetura baseada em serviços para aplicações de Redes Sem Fio que utilizam tecnologias de Redes de Sensores e Atuadores Sem Fio, focando no contexto de sistemas utilizados para monitoramento remoto, atuação e segurança patrimonial, embora essa arquitetura proposta necessite ser flexível o suficiente

para ser migrada a outro domínio de aplicação sem grandes alterações. Essa Rede Sem Fio é uma rede interna e privada, mas deve fornecer uma interface através de uma rede pública, como a Internet, por exemplo, de forma a permitir que o usuário possa acessá-la remotamente através de dispositivos móveis (como *Smartphones* ou telefones celulares) ou fixos (uma estação de trabalho, por exemplo), desde que estes dispositivos também estejam conectados de alguma maneira à mesma rede pública. É necessário fornecer ao usuário uma forma amigável, eficiente, estável e segura de acessar sua rede.

Para um suporte adequado a este ambiente é importante que a infraestrutura do nodo *gateway* seja complementada com alguns componentes para dar suporte ao modelo de serviços. Esta infraestrutura oferece novas funcionalidades relacionadas a SOA.

Um diagrama preliminar com a visão funcional da arquitetura e seus componentes é ilustrado na figura 2.

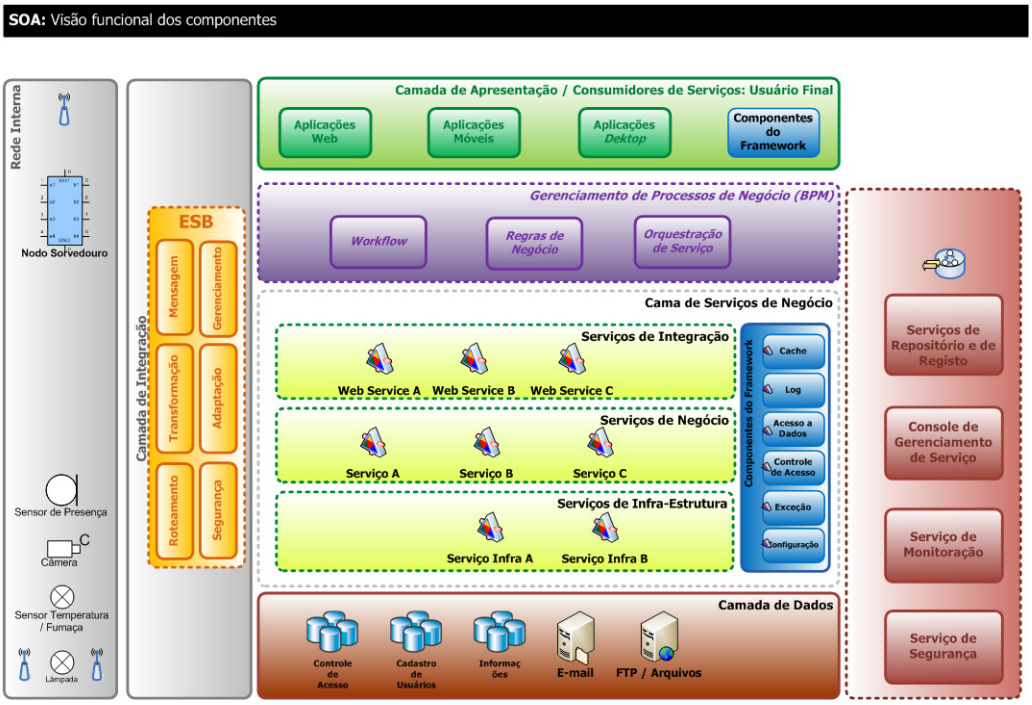

Figura 2: Visão Funcional da Arquitetura Orientada a Serviços

Este ambiente contém os seguintes componentes principais:

- Um ESB (*Enterprise Service Bus*) que fornece o mecanismo de processamento de mensagens e todo suporte para o transporte, roteamento, segurança, adaptadores e transformação das mensagens de comunicação. Este componente

oferecerá o suporte necessário para integração de aplicações em diferentes plataformas ou através de diferentes protocolos;

- Um serviço de registro e metadados (*Serviço de Registro*) que é usado para localizar as informações sobre os serviços publicados no ambiente;
- Um componente para publicação e descoberta de serviços (*Serviço de Repositório*) com suporte aos padrões do mercado como, por exemplo, UDDI. O UDDI é um padrão específico para publicação e pesquisa de serviços;
- Um mecanismo para orquestração e automação de processos de negócio (BPM – *Business Process Management*). Por exemplo, orquestração de serviços ou workflow;
- Um mecanismo (*Business Rules Engine*) de execução e gerenciamento de regras de negócio;
- Um mecanismo de gerenciamento de serviço que permita controlar os vários aspectos dos serviços;
- Um mecanismo de segurança que permita controlar os vários aspectos de segurança dos serviços.

O ESB, embora isolado na camada de integração, não significa que seja utilizado somente na comunicação com os componentes desta camada. Está agrupado na camada de integração devido ao papel que desempenha na arquitetura, porém também será usado para comunicação dos serviços e seus consumidores das outras camadas. Ideal para cenários onde os consumidores foram construídos com tecnologias diferentes ou demandam alguma transformação de dados ou de protocolo.

O gerenciamento de identidade se refere à autenticação das identidades e o gerenciamento dos seus atributos, tais como papéis e privilégios.

A segurança diz respeito à autenticação, autorização, integridade, confidencialidade, auditoria e privacidade.

Assim, ao final deste trabalho de pesquisa, espera-se propor a definição de uma arquitetura e uma infraestrutura capazes de permitir que um usuário consiga controlar e monitorar o funcionamento de dispositivos eletro/eletrônicos remotamente e que seja capaz de coletar variados tipos de informações do ambiente local da sua rede privada, tais como temperatura, presença de determinado gás, detecção de movimento, detecção de fumaça, detecção de abertura de uma janela ou porta, etc. Além de todos os pré-requisitos técnicos, a solução proposta deverá apresentar custo financeiro menor do que outras soluções já apresentadas ou comercializadas.

A figura 3 ilustra o ambiente utilizado no estudo de caso, uma Rede Sem Fio para atuação e monitoração com fins de vigilância patrimonial.

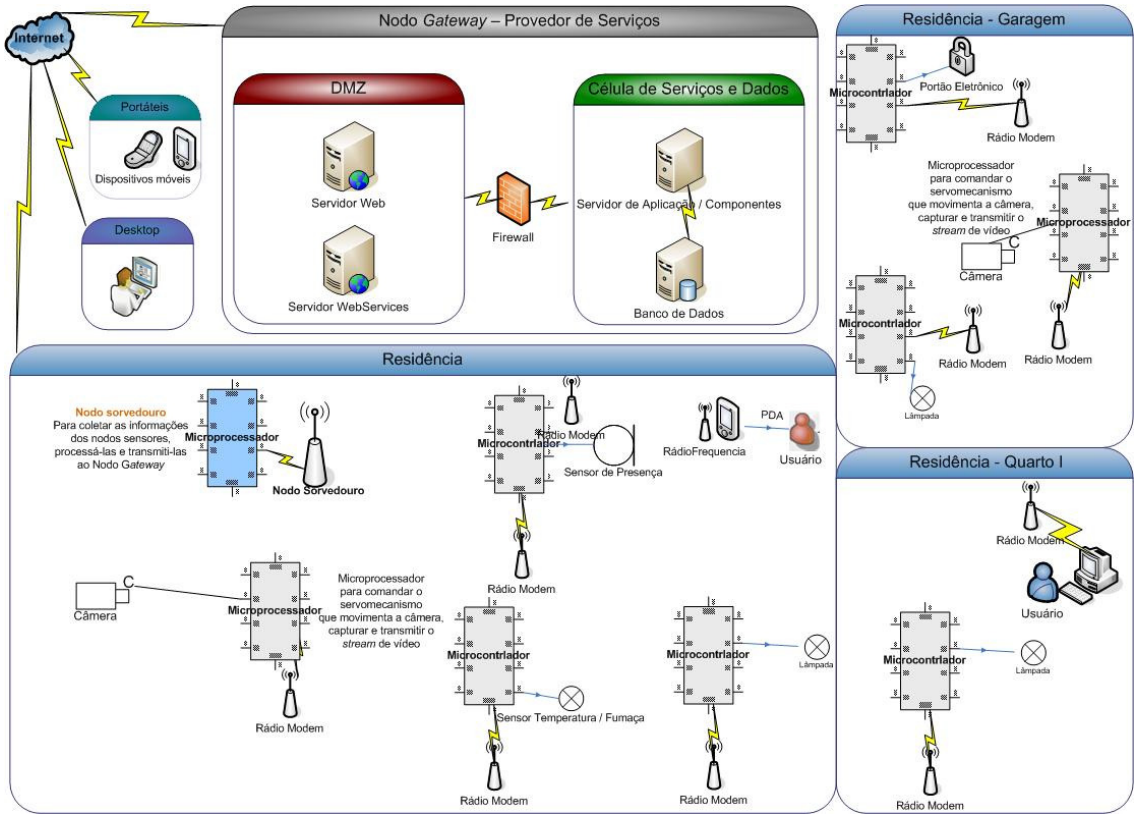

Figura 3: Uma Rede Sem Fio para atuação e monitoração de uma residência

Neste sistema foi construída uma infraestrutura para permitir que um usuário interaja com sua rede interna sem fio (por exemplo, acendendo ou apagando uma luz, ligando um dispositivo eletrônico, etc.) e receba eventos disparados por gatilhos pré-programados (como por exemplo, detecção de intrusão, abertura de uma janela ou porta, detecção de fumaça ou gás, etc.) através de qualquer dispositivo móvel ou PC conectado à Internet. Nesse ambiente, o usuário deve acessar o provedor de serviços, que faz o papel de nodo *gateway*, de forma a ter acesso aos recursos de sua rede privada. Como o nodo *gateway* deve possuir funcionalidades que exigem mais recursos computacionais, resolvemos criar uma célula especial para alocá-lo, chamada de Provedor de Serviços.

Quando o usuário acessar o Portal através de um navegador, ele terá que digitar seu usuário e senha cadastrados no sistema para se autenticar. Após a autenticação, ocorrerá o processo de autorização, onde serão verificados os serviços que esse usuário tem a sua disposição, ou seja, os dispositivos que ele tem instalado em sua rede interna.

Após o usuário ser autenticado e autorizado e ter finalmente recebido seu *token,* ele será direcionado para a tela de controle, gerenciamento e monitoração de sua rede. De acordo com os serviços que foram autorizados (as transações disponíveis para o usuário), um menu dinâmico será gerado que o permitirá realizar operações remotamente, através de seu navegador.

Uma figura ilustrando um protótipo da tela apresentada ao usuário, onde é possível monitorar remotamente um ambiente através de uma câmera de vigilância com movimentos nos eixos X e Y (movimentos *pan* e *tilt*), é mostrada a seguir:

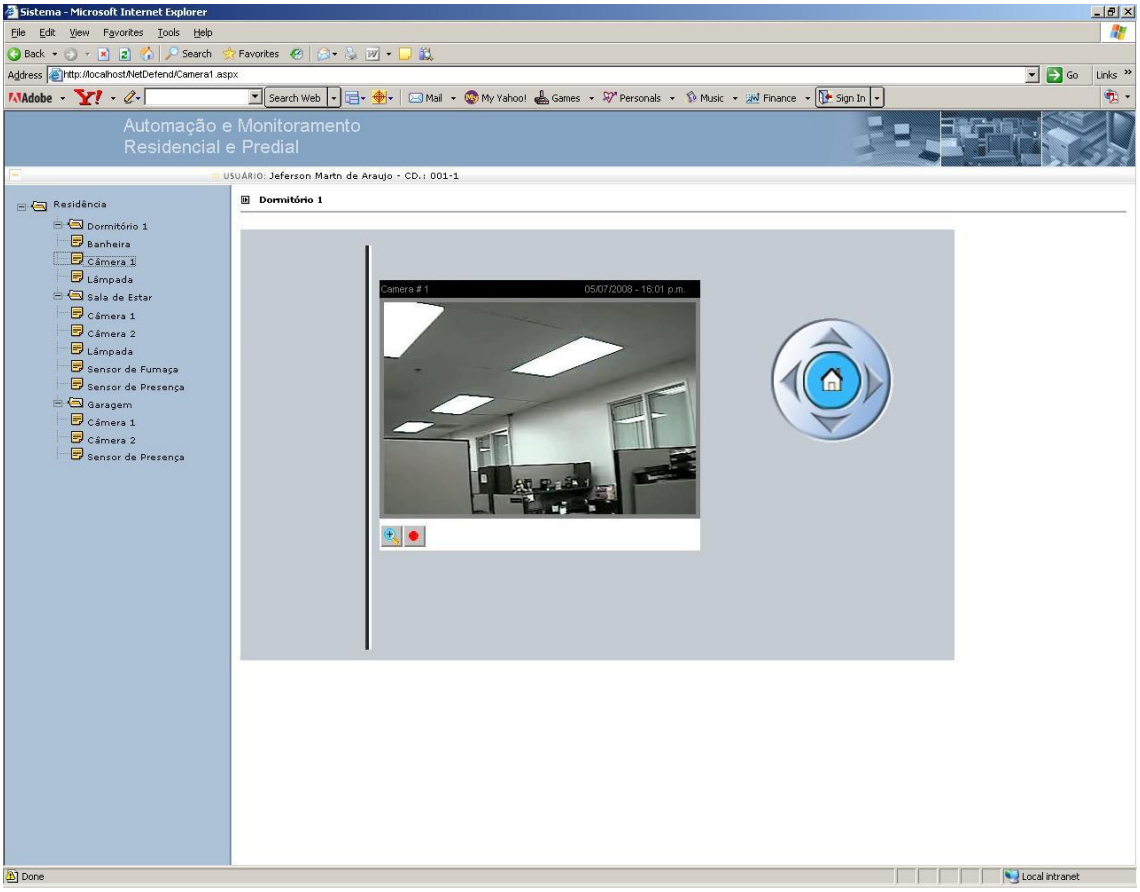

Figura 4: Interface Web para monitoramento remoto

1.3 Motivação e Relevância

No campo das RSASF, muitos assuntos têm sido abordados e inúmeras pesquisas tem sido realizadas na área de segurança, desempenho dos microprocessadores, otimização do consumo de energia e do alcance nas transmissões, protocolos de roteamento, tamanho dos nodos sensores, etc. No entanto, um dos maiores problemas e desafios das RSASF - a dificuldade no desenvolvimento de aplicações - ainda persiste.

Pesquisas recentes mostram que modelos de programação orientada a serviços podem contribuir para agilizar e facilitar a comunicação e interoperabilidade entre dispositivos heterogêneos. Uma RSASF com arquitetura orientada a serviços permite aos desenvolvedores rapidamente desenvolver aplicações compostas, reutilização de serviços, etc.

Atuação e monitoração predial bem como segurança patrimonial (*wide-area surveillance)* são importantes aplicações das RSASF com enorme potencial comercial. No entanto, faz-se necessário um profundo estudo de tecnologias recentes do campo da Ciência da Computação e da Engenharia Elétrica a fim de agregar valor à solução final através do conhecimento adquirido com estas pesquisas.

No entanto, a utilização de tecnologias das RSASF para atuação e monitoração de residências tem sido tratada como um assunto futurístico, já que a alta complexidade tecnológica que a solução pode exigir implica em altíssimos custos financeiros, restringindo, assim o uso dessa tecnologia. Em vista disso, este estudo tem sido motivado pela possibilidade de combinar infraestrutura e arquitetura de *hardware* e *software* a fim de se obter uma solução de baixo custo.

O mercado desses serviços é promissor, pois este setor de interconexão de dispositivos eletrônicos através de comunicação digital está sendo cada vez mais necessário devido aos benefícios oferecidos e relacionados à atuação, monitoração e sensoriamento remotos (via Internet, por exemplo).

1.4 Estrutura do Trabalho

Este trabalho consiste de sete capítulos, explicados seqüencialmente em seguida. Neste primeiro capítulo, “Introdução”, apresentamos em linhas gerais o contexto do problema e a área de pesquisa que iremos trabalhar e em seguida listamos nossos principais objetivos, a motivação e relevância do projeto. No segundo e terceiro capítulos, focamos em apresentar e discorrer sobre a fundamentação teórica essencial para o entendimento dos capítulos subseqüentes.

Após esta apresentação dos conceitos fundamentais, seguimos para o quarto capítulo, com o objetivo de fazer um mapeamento e levantamento realista dos principais serviços de uma RSASF e a metodologia para a construção de uma arquitetura orientada a serviços. Nesse mesmo capítulo também apresentamos o framework que foi desenvolvido para prover serviços genéricos de infraestrutura. Considerando as etapas da metodologia descrita no quarto capítulo e os estudos do segundo e terceiro capítulos, seguimos na apresentação da arquitetura candidata no quinto capítulo, explicando e justificando todas as atividades que foram desenvolvidas para a obtenção desta arquitetura. Em seguida, um detalhamento do desenvolvimento de uma aplicação prática (estudo de caso) baseada em conceitos de RSASF e no paradigma arquitetural SOA, é realizado no sexto capitulo.

Em seguida, o sétimo capítulo sintetiza os principais resultados, parciais e finais, obtidos neste trabalho, tecendo uma explicação elucidativa sobre os mesmos e apresentando, por fim, a conclusão sobre tudo o que foi abordado.

CAPÍTULO 2 – FUNDAMENTAÇÃO TEÓRICA

As Redes de Sensores e Atuadores Sem Fio constituem um domínio da computação distribuída e têm sido alvo de grande interesse de pesquisa nos últimos anos. Uma RSASF é um tipo de rede *ad-hoc* com uma série de características e requisitos peculiares. As próximas seções fornecem conceitos básicos sobre RSASF.

2.1 Visão Geral dos Principais Conceitos das RSASF

Devido aos avanços tecnológicos na comunicação sem fio e na eletrônica nos últimos anos, como o desenvolvimento de redes de telecomunicações com enlace sem fio de baixo custo e alta largura de banda e do baixo consumo de energia pelos dispositivos microeletrônicos, as redes de sensores vêm recebendo atenção cada vez maior por parte da comunidade científica e da sociedade em geral. Os nodos sensores dessas redes alcançaram tamanhos bastante reduzidos e atualmente são capazes de capturar e processar dados e comunicar com nodos vizinhos através de um canal de RF (rádio freqüência), formando as Redes de Sensores e Atuadores sem Fio, ou RSASF. Uma RSASF é projetada para detectar eventos ou fenômenos, coletar e processar dados, e transmitir essa informação coletada aos usuários interessados. As características básicas de uma RSASF são:

- *Mudanças freqüentes na topologia e Capacidade de auto-organização*: Sensores numa RSASF podem ser perdidos devido à sua destruição física ou falta de energia. Eles também podem ficar incomunicáveis devido a problemas no canal de comunicação ou por decisão de um algoritmo de gerenciamento da rede. Este fato pode acontecer por diversas razões como, por exemplo, para economizar energia ou devido à presença de outro sensor na mesma região que já coleta o dado desejado. A situação contrária também pode acontecer: sensores inativos se tornarem ativos ou novos sensores passarem a fazer parte

da rede. Em qualquer um dos casos é necessário haver mecanismos de auto-organização para que a rede continue a executar a sua função [17].

- *Limitação de energia, de memória para processamento ou armazenamento, de poder de transmissão e de computação nos nodos sensores:* os nodos sensores são extremamente limitados em energia, memória e poder de processamento. Qualquer estudo de arquitetura, de protocolo de rede, etc. devem levar em consideração estes fatores.

- *Tarefas colaborativas, agregação dos dados coletados e grande quantidade de nodos sensores na rede:* algumas aplicações das RSASF podem exigir a existência de até 100 mil sensores, como por exemplo, as aplicações ambientais para monitoramento de florestas. Numa RSASF, os nodos sensores devem executar alguma tarefa colaborativa, onde é importante detectar e estimar eventos de interesse e não apenas prover mecanismos de comunicação. Devido às restrições das RSASF, normalmente os dados são agregados para reduzir o número de mensagens que precisam ser transmitidas por ela. Os dados coletados são combinados e sumarizados ainda na rede, pelo nodo sorvedouro, antes de serem enviados ao nodo *gateway* [17].

2.2 Componentes das Redes de Sensores Sem Fio

Os principais componentes das redes de sensores são os nodos sensores e atuadores (geralmente micro-controladores com poder de processamento e memória muito limitados), nodos sorvedouros (microprocessadores), e porventura, nodos *gateways* para comunicação com redes externas como, por exemplo, a Internet.

2.2.1 Os nodos sensores/atuadores

Nodos sensores/atuadores são dispositivos autônomos equipados com capacidades de sensoriamento, processamento, comunicação e de atuação no ambiente. Quando estes nodos são dispostos em rede de comunicação sem fio, formam as Redes de Sensores e Atuadores Sem Fio. Esses nodos coletam dados via seus sensores, processam localmente ou coordenadamente e enviam as informações para um ou mais sorvedouro [17]. Um nodo na rede tem uma tarefa específica, de acordo com a especialização do seu sensor, tal como telemetria, pressão, temperatura, sensores de presença, fumaça, etc. Cada sensor normalmente cobre uma pequena área, da ordem de 20 metros, aproximadamente, tendo como características serem bastante compactos. Um nodo sensor/atuador pode ser descrito de acordo com o diagrama em blocos da figura 5.

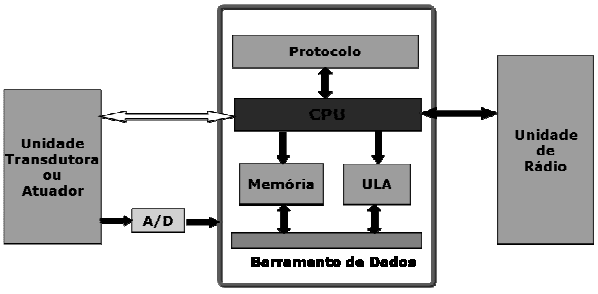

Figura 5: Componentes funcionais de um nodo sensor/atuador

Nos nodos sensores, os dados fluem unidirecionalmente da Unidade Transdutora, passam por um conversor Analógico/Digital e chegam à Unidade Controladora. O consumo de energia do Transdutor depende do modo de operação e do tipo de grandeza medida.

A Unidade de Controle contém:

- CPU e memória (a CPU costuma variar entre uma arquitetura de 8 a 16 bits com *clock* variando entre 4 a 40 MHz e a memória geralmente varia entre 128Kbytes a 40 Mbytes, ambos possuindo baixo consumo de energia);

- Unidade Lógica e Aritmética (ULA);

- Opcionalmente, uma unidade de protocolo, para implementar as funções de protocolo de rede.

A Unidade de Controle, no nodo sensor, geralmente é um micro-controlador. A figura abaixo ilustra um micro-controlador da família PIC 16F84a [18] com uma CPU, memória e ULA.

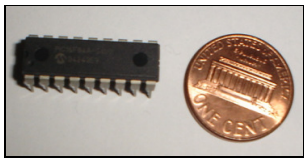

Figura 6: Micro-controlador representando a unidade de controle de um nodo sensor

O consumo da Unidade de Controle depende da velocidade do relógio (quanto menor a freqüência menor o consumo) e do modo de operação. O consumo pode ser medido pelo número de ciclos de relógio para diferentes tarefas como o processamento de sinais, verificação de código de erro, etc.

A Unidade de Rádio provê o protocolo MAC, que gerencia a transmissão de dados pelo link sem fio. As figuras a seguir ilustram duas unidades de rádio que podem ser utilizadas para prover comunicação sem fio a um nodo sensor:

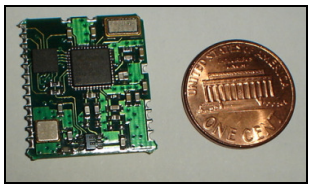

Figura 7: Módulo de comunicação por RF da Radiotronix [19], modelo Wi.232 DST-R

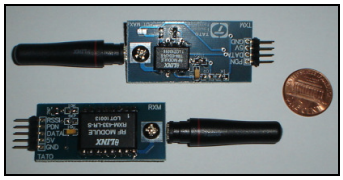

Figura 8: Figura 8: Módulo de comunicação por RF da Tato [20], modulo TX e RX separados

Para RSASF, uma fonte de energia contínua e disponível no ambiente é de grande importância. Os nodos sensores são alimentados por uma fonte de energia geralmente bastante limitada. Em muitas aplicações, os sensores são colocados em áreas remotas, não permitindo o fácil acesso a esses elementos para manutenção. Neste cenário, o tempo de vida de um sensor depende da quantidade de energia disponível. Aplicações, protocolos, e algoritmos para RSASF devem ser escolhidos criteriosamente, não tendo como parâmetros apenas sua eficiência, funcionalidades e capacidade, mas também a quantidade de energia consumida. Assim, o projeto de qualquer solução para esse tipo de rede deve também levar em consideração o consumo, o modelo de energia e o mapa de energia da rede.

O modelo de energia representa os recursos físicos de um sensor, que consomem energia e interagem com um modelo de funções. O modelo pode ser visto como um provedor de energia para elementos consumidores, que depende de uma bateria que tem uma capacidade finita de energia armazenada.

Existem diferentes tecnologias de fabricação para essas baterias, referindo-se ao consumo de energia. A escolha da bateria a ser utilizada nos nodos sensores deve considerar algumas características como volume, condições de temperatura e capacidade inicial.

2.2.2 Os nodos sorvedouros

São nodos especiais, geralmente microprocessadores, que captam os pacotes dos diversos nodos sensores, encapsulam os dados coletados e enviam a um servidor *gateway*. O enlace de dados entre o *gateway* e o nodo sorvedouro possui maior largura de banda do que enlace do sorvedouro com os nodos sensores, podendo até mesmo utilizar-se de um meio físico como fibra óptica ou cabeamento estruturado.

Os nodos sorvedouros fazem o processamento das informações coletadas em função da aplicação. Os dados coletados poderão estar sujeitos a compressão, correlação, criptografia, assinatura digital, etc. Outro processamento importante diz respeito aos gatilhos que definem os estímulos para a coleta dos dados. Por exemplo, os nodos sensores de temperatura podem ter seu processamento estimulado em função de uma variação ou ao rompimento dos limites estabelecidos ou através de um gatilho disparado por um comando vindo do servidor *gateway*.

A figura 9 ilustra um microprocessador, fabricado pela *Rabbit Semiconductors* [21], candidato ao papel de nodo sorvedouro em uma RSASF, com uma saída Ethernet, útil para ser utilizada na comunicação com o servidor *gateway*.

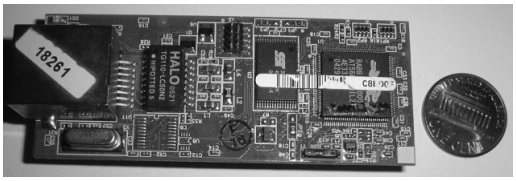

Figura 9: Figura 9: Microprocessador da família *Rabbit, Série 3000*

2.2.3 Os nodos gateways

São equipamentos que fazem a interface dos dados recebidos pelo nodo sorvedouro com uma aplicação cliente final. O *gateway* pode disponibilizar as informações coletadas de diversas formas, como por exemplo, através de um servidor de páginas Web, através de uma arquitetura cliente/servidor utilizando um *socket* TCP ou até mesmo através de uma interface *WebService* [22].

Assim, as interações entre as aplicações disponibilizadas aos usuários e a RSASF são feitas através do nodo *gateway*. Esse nodo deve fornecer um arcabouço ou *framework* capaz de fornecer as interfaces para o desenvolvimento das aplicações a serem utilizadas pelos usuários e as APIs (*Application Programming Interface*) de programação de baixo nível para comunicação com o nodo sorvedouro [23].

2.3 Aplicações das Redes de Sensores Sem Fio

O recente desenvolvimento de microprocessadores e micro-controladores de alto desempenho e os avanços nos dispositivos sensores têm estimulado o interesse no desenvolvimento de "sensores inteligentes" – sensores de grandezas físicas, químicas ou biológicas combinados com circuitos integrados. Não é rara a utilização de vários tipos de sensores combinados em um único *chip*, com o circuito integrado desse *chip* controlando todos estes sensores.

As aplicações das RSASF são inúmeras e descrever todas as suas utilizações é uma tarefa impossível de ser realizada. Entretanto, alguns domínios de aplicação podem servir de exemplo e ajudar a justificar o crescente interesse nesse tipo de tecnologia. Alguns desses domínios de aplicação são discutidos nas seções seguintes.

2.3.1 Mineração

Monitoração em minas de carvão no subsolo é uma importante aplicação das RSASF com potencial comercial. A utilização de cabeamento para conectar os nodos sensores a um servidor de processamento dos dados coletados requer uma extensa instalação de cabos, o que se acaba tornando complexo devido às pobres condições de trabalho no subsolo e ao alto custo de manutenção deste cabeamento. Além do mais, a comunicação via cabo torna o sistema menos escalável, na medida em que o túnel cresce e demanda a instalação de novos pontos sensores e, conseqüentemente, mais cabeamento. Nesse caso, um sistema de comunicação sem fio agrega a vantagem de instalações e ajustes mais flexíveis. Devido à inerente interferência na comunicação causada pela convivência com máquinas mineradoras,

torna-se impossível manter um canal de comunicação sem fio diretamente entre os nodos sensores e o servidor, tornando indispensável o desenvolvimento de um protocolo de roteamento alternativo. A natureza instável das construções geológicas em minas de carvão torna os túneis propensos a alterações estruturais, sendo que essa instabilidade pode resultar em acidentes fatais causados pelos desmoronamentos. Dessa forma, é essencial a instalação de uma RSASF de forma a permitir que os nodos sensores detectem uma área com principio de desmoronamento e comuniquem ao nodo sorvedouro tal ocorrência. Em vista destes cenários, Li et. al [24] projetaram um sistema de sensores chamado SASA (*Structure-Aware Self Adaptative*) que visa endereçar dois pontos cruciais na monitoração do subsolo de minas de carvão: 1) habilidade de rapidamente detectar o desmoronamento de uma área (através de um algoritmo de detecção de buracos nas superfícies internas dos túneis) e reportar ao nodo sorvedouro tal ocorrência; e 2) habilidade de manter a integridade do sistema mesmo quando a estrutura da rede de sensores é alterada, detectando e re-configurando nodos sensores inativos. O sistema SASA pode fornecer outras funções tais como detecção de gás e água e monitoração da densidade de oxigênio. No estudo de Li, foi construído um protótipo com 27 *motes Mica2* [25] usando o *TinyOS* para ser instalado na mina de carvão D.L. (uma das minas de carvão mais automatizadas, mantendo o segundo lugar em produção mundial), usada como um estudo de caso. Os experimentos mostraram que o SASA conseguiu detectar e indicar a formação de buracos na superfície onde 80% desses buracos estavam localizados dentro de no máximo 1 metro de sua posição real e 99% estava a menos de 2 metros. Baseado nos dados coletados experimentalmente, Li realizou simulações para predizer o desempenho, a escalabilidade e estabilidade do SASA.

2.3.2 Monitoração ambiental

Mainwaring et. al [26] apresenta um estudo detalhado sobre utilização de RSASF para monitoração ambiental. Nesse estudo, os autores apresentam um conjunto de requisitos, restrições e diretrizes que servem como base para definir a arquitetura de muitas aplicações neste domínio. O estudo descreve também os componentes básicos para uma rede

de sensores neste domínio de aplicação – os sensores e as plataformas de *hardware*, as características da rede de comunicação, a interconexão dos nodos e a flexibilidade e habilidade para gerenciamento dos dados. As redes de sensores representam um avanço significativo sob os métodos tradicionais de monitoração, considerados invasivos. Os sensores podem ser instalados antes de um período característico (por exemplo, no caso de animais, antes do período de acasalamento) ou enquanto as plantas estão dormentes ou o solo congelado (no caso de estudos botânicos), de modo a não causar distúrbios no ambiente. Os sensores também podem ser instalados em locais inseguros para estudos de campos repetitivos.

2.3.3 Agropecuária

As RSASF podem também ser utilizadas na agropecuária a fim de se obter um rastreamento e controle autônomo de animais. Por exemplo, Wark et. al [27] estudou o uso de uma RSASF na pecuária com o objetivo de prevenir brigas entre búfalos em fazendas de procriação. Esta é uma aplicação importante na pecuária, na medida em que brigas de animais de grande porte em época de procriação podem resultar em grandes perdas financeiras para os produtores. Para evitar tais brigas, o estudo de Wark propôs um sistema que é capaz de determinar a localização em tempo real dos búfalos e aplicar um estímulo elétrico apropriado quando um búfalo se aproximar de outro ou então se aproximar de um cercado reprodutivo que não seja o dele. A figura 10 ilustra um búfalo usando um colar com um atuador e antenas para comunicação com a rede, de forma a separá-los territorialmente, sem intervenção humana, em cercados reprodutivos.

Figura 10: Foto tirada de um búfalo usando um colar com hardware, antenas e bateria [27]

2.3.4 Médica

Embora a técnica de utilização de sensores na medicina seja relativamente nova, o número de propostas de aplicações interessantes é grande e tem aumentado. A fim de evidenciar o potencial de utilização das RSASF na biomedicina e alavancar a pesquisa nessa área, Schwiebert et. al [28] levantou algumas possíveis aplicações, as quais serão brevemente descritas a seguir:

2.3.4.1 Retina Artificial

Nesta seção, apresentamos uma prótese de retina desenvolvida com o projeto *Smart Sensors and Integrated Microsystems* (SSIM) da *Wayne State University* e o *Kresge Eye Institute*. Um dos objetivos deste projeto é construir uma retina artificial que permita deficientes visuais (sem visão alguma ou com visão limitada) passarem a possuir "determinado nível de visão". Dessa forma, foi projetado um *microchip* com um circuito integrado e um vetor de micro sensores. Uma versão desse dispositivo implantado em um olho humano pode ser observada na figura 11.

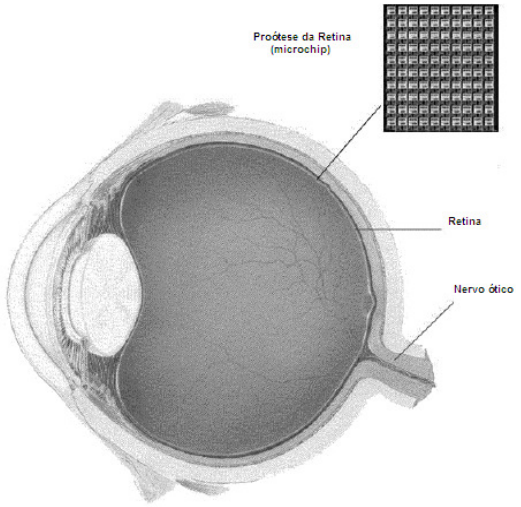

Figura 11: Localização dos sensores dentro da retina [28]

Como ilustrado na figura 11, o lado frontal da retina está em contato com o vetor de micro sensores. As transmissões dentro do olho fluem da seguinte forma: o lado traseiro da retina é estimulado eletricamente (pela prótese artificial) pelo chip. Estes sinais elétricos são convertidos em sinais químicos pela ganglia basal e outros tecidos e são então transmitidos ao cérebro pelo nervo ótico A transmissão dos sinais a partir dos sensores implantados no olho ocorre de forma similar, mas no sentido oposto. Os sinais neurológicos do ganglia são capturados pelos micros sensores e o sinal e a intensidade relativa são transmitidos para fora do chip. O propósito desse fluxo reverso dos dados é para determinar o mapeamento entre a imagem de entrada e os sinais neurológicos resultantes que nos permitirão enxergar aquela imagem. Não é viável realizar o processamento interno deste mapeamento usando os recursos computacionais limitados do *chip* implantado. Sendo assim, é necessário estabelecer uma forma de comunicação sem fio entre o chip e um dispositivo externo.

2.3.4.2 Monitores do Nível de Glicose

Nodos sensores com caráter biomédico podem ser implantados em pacientes e transmitir o nível de glicose no sangue a um dispositivo externo através de uma comunicação sem fio. Esta técnica, menos invasiva, permite que o paciente verifique o nível de glicose no

sangue com maior freqüência durante o dia e ainda com maior precisão, permitindo ao paciente tomar ações corretivas com maior antecedência do que através dos métodos normais de verificação do nível de glicose.

2.3.4.3 Monitores de órgãos

Infelizmente, mais do que um terço dos corações e fígados disponíveis para transplantes são descartados devido à pequena janela de tempo que um órgão pode sobreviver antes de ser transplantado. Evidentemente, a possibilidade de um melhor entendimento das condições do órgão evitaria tal desperdício. Alguns pesquisadores estão estudando a possibilidade de incluir sensores nos órgãos para monitorar os níveis de determinados gases nesses órgãos, tais como oxigênio, metano, dióxido de carbono, etc. a fim de se determinar a viabilidade do transplante.

2.3.4.4 Detectores de Câncer

Nodos sensores de caráter biomédico podem ter um papel fundamental na detecção do câncer em fase inicial, aumentando sobremaneira a possibilidade de cura. É sabido que células cancerígenas expelem óxido nítrico, que afeta o fluxo de sangue na região em volta do tumor. Sensores com capacidades de detectar estas alterações no fluxo do sangue podem ser posicionados em locais suspeitos. Dessa forma, qualquer anomalia pode ser notada com maior antecedência.

2.3.5 Segurança

Câmeras de monitoramento estão sendo cada vez mais utilizadas, e vêm permeando ambientes públicos e privados, para aumentar a segurança pública ou monitorar tráfego de veículos e pedestres, no esforço de coibir e prevenir atos criminosos. O *Eletronic Privacy Information Center* [29] estima que cerca de 1.5 milhão dessas câmeras estejam instaladas no Reino Unido e 2 milhões nos Estados Unidos [30]. Esses sistemas de segurança são de propriedade de empresas públicas ou privadas e se beneficiam dos avanços tecnológicos e das sofisticações dos sistemas computacionais enquanto o custo de instalação e manutenção vem decrescendo exponencialmente.

No litoral sul do Estado de São Paulo, cidades como Praia Grande, Guarujá e Santos foram as pioneiras a adotarem esse serviço de segurança permanente. Os locais para instalação dessas câmeras foram selecionados de acordo com a incidência da criminalidade.

De acordo com dados da Policia Militar, após a instalação das câmeras e do sistema de monitoramento digital, houve uma redução de cerca de 30% nos crimes registrados nos locais de abrangência da monitoração dessas câmeras. A capital do Estado de São Paulo, uma das metrópoles mais populosas do mundo, também está investindo no monitoramento para segurança publica e patrimonial. Até o final de 2007 a cidade de São Paulo possuía 99 câmeras com capacidade de movimentar-se 360º em torno do seu eixo, instaladas de maneira que seja monitorado o maior número de ruas possível [31].

Balancear a necessidade de segurança, conforto e eficiência disponibilizados por esses mecanismos de vigilância com a necessidade de privacidade é um grande desafio. Um estudo interessante que garante o direito a indivíduos de proibir a divulgação de vídeos contendo suas imagens vem sendo desenvolvido por J. Brassil [76]. Nesse estudo, o autor desenvolveu o sistema *Cloak,* que é capaz de aumentar a privacidade dos cidadãos sem necessidade de alterar os sistemas digitais de vigilância existentes atualmente. Outro projeto relativo à privacidade versus vigilância é o *NYC Surveillance Câmera Project* [32], que possui uma compilação das câmeras instaladas em Manhattan e fornece um mapa com o local de instalação e informações detalhadas sobre as finalidades de cada câmara, tais como monitoração de trânsito, pedestres, segurança, etc.

.

2.3.6 Militar

Interesse militar em redes de sensores e atuadores sem fio é motivado pelos muitos problemas que podem ser resolvidos com segurança e eficácia usando sensores. Por exemplo, uma RSASF poderia ser instalada em um campo de batalha para rastrear movimentos de tropas. Sensores embutidos em pequenos robôs poderiam conduzir a detecção de minas. Ainda com fins militares, uma RSASF poderia ser utilizada para detectar o uso de armas químicas ou biológicas e, via comunicação sem fio, reportar a presença destas armas para a proteção de tropas.

2.4 Atuação e Monitoramento Predial utilizando os conceitos das RSASF

Atuação e Monitoramento Predial representam uma classe de aplicações dos conceitos de Redes de Sensores e Atuadores sem Fio com enorme potencial de benefícios para a comunidade cientifica e a sociedade em geral.

Nos últimos anos as crescentes preocupações com segurança, custos e impactos ambientais levaram empresas e pessoas a buscarem soluções alternativas e inovadoras proporcionando um significativo avanço no campo da monitoração/atuação predial e residencial e despertando o interesse das pessoas.

Por outro lado, o aumento de escala, a redução de custos, o desenvolvimento da computação pessoal e da Internet, trouxeram ao cotidiano das pessoas e empresas funcionalidades que antes só eram vistas em filmes, tais como câmeras acessadas e controladas por dispositivos conectados à Internet, acionamento remoto de aparelhos (luzes, irrigação, segurança, alarmes, câmeras, portas e aparelhos eletrônicos em geral). Essa capacidade de suportar interações remotas é um aspecto essencial para a monitoração e segurança patrimonial.

Uma dos grandes desafios na atuação e monitoramento predial está no caráter interdisciplinar, envolvendo vários conceitos de ciências como Arquitetura, Engenharia, Ciência da Computação [33]. Além disso, deve existir um determinado grau de flexibilidade na arquitetura da solução de forma a permitir um rápido e eficiente desenvolvimento das aplicações e a integração de diferentes equipamentos e dispositivos eletrônicos. A figura 12 ilustra os setores que essa arquitetura e infraestrutura devem considerar.

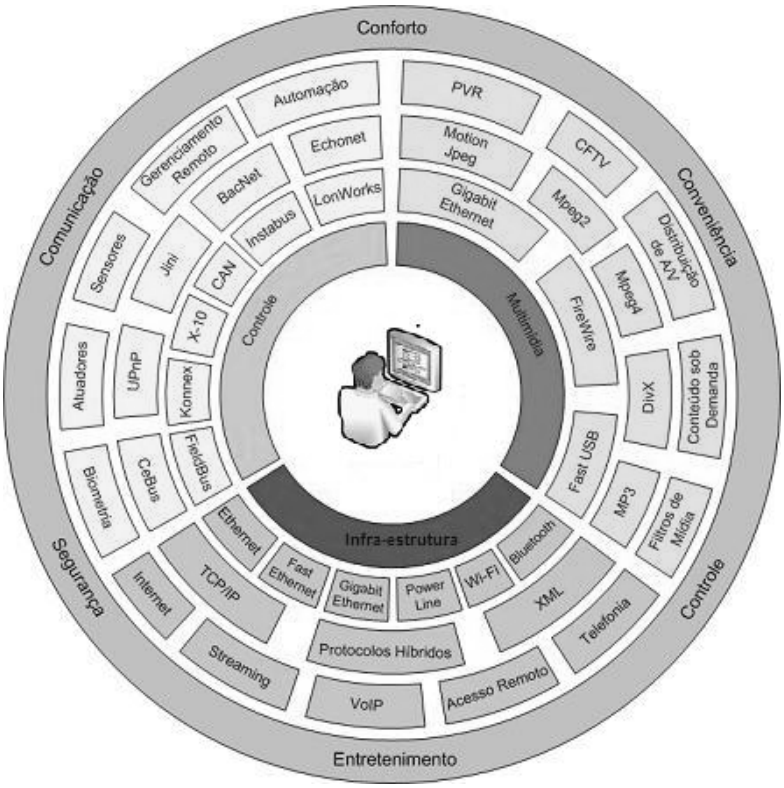

Figura 12: Tecnologias envolvidas num sistema de atuação, segurança e monitoramento patrimonial, adaptado de [34]

O setor mais importante é o setor de controle, responsável pelo gerenciamento dos atuadores, dispositivos de controle e sensores. Neste setor encontram-se os nodos sensores, nodos sorvedouros e o nodo gateway. O setor de infraestrutura é o responsável por interconectar os nodos da Rede Sem Fio. Por fim, temos o setor multimídia, responsável pela captura de informações de áudio e vídeo, pela forma de codificação do *stream* com determinada qualidade de serviço assegurada e pela interface de apresentação dos dados ao usuário, entre outros.

Todas estas motivações evidenciam o crescimento da demanda pelos usuários e o conseqüente crescimento de investimentos em pesquisas por centros acadêmicos de excelência e por grandes corporações.

CAPÍTULO 3 – ARCABOUÇOS DE DESENVOLVIMENTO E PARADIGMAS ARQUITETURAIS

Neste capítulo são abordados os conceitos básicos sobre *frameworks* de desenvolvimento e apresentados os dois principais *frameworks* disponíveis para o desenvolvimento de aplicações. Em seguida, são apresentados os conceitos de "Padrões Arquiteturais", "Modelos de Referência" e "Arquiteturas de Referência" que irão nortear o restante deste trabalho.

3.1 Arcabouços de Desenvolvimento (Frameworks)

Definem-se arcabouços de desenvolvimento ou *frameworks* como um conjunto de classes-base (esqueleto) sobre o qual um sistema é construído. Silva [35] define *frameworks* como um conjunto de estrutura de classes que constituem implementações incompletas que quando estendidas, permitem produzir diferentes artefatos de um sistema. Larman [36] define *frameworks* como um sistema extensível para um conjunto de sistemas relacionados. Também é possível ser classificado como um conjunto coeso de classes que colaboram para fornecer serviços para o núcleo invariante de um sistema lógico. É composto de classes abstratas, concretas e provê um modelo de interação ou colaboração entre as instâncias de classes definidas pelo *framework*.

Assim, um *framework* é uma coleção de classes, interfaces e padrões dedicados a resolver uma classe de problemas através de uma arquitetura flexível. A diferença de um *framework* e uma biblioteca de classes está no fato de que na biblioteca as classes são únicas e independentes das outras. Já no *framework* as dependências entre as classes estão embutidas em si. No *framework*, o conjunto de classes deve ser flexível e extensível (ou seja, o *framework* deve conter funcionalidades abstratas que poderão ser implementadas).

Atualmente, os dois *frameworks* ou plataformas para desenvolvimento mais amplamente utilizados são o *Framework .NET* e o *Framework Java*. Uma breve introdução a essas duas plataformas de desenvolvimento será feita em seguida.

3.1.1 Plataforma .NET

O termo *Framework .NET* [37] foi o nome dado a vários componentes e serviços que foram combinados para criar um rico ambiente de desenvolvimento. Ele inclui uma grande quantidade de classes (mais de seis mil) que fornecem a maioria das funcionalidades anteriormente existentes na API (*Application Programming Interface*) do Microsoft Windows. Além disso, o *Framework .NET* possui o *Common Language Runtime*(*CLR*), que é o responsável por executar o código da plataforma *.NET*. As aplicações desenvolvidas utilizando linguagens *.NET* são compiladas para a *Microsoft Intermediate Language (MSIL)* através da IDE de desenvolvimento. Em seguida, ela é convertida pelo CLRem código nativo quando é executada pela primeira vez. O resultado final é o benefício do desempenho de um código totalmente compilado, e não de um que seja interpretado no tempo de execução.

Todas as linguagens que têm suporte do CLR usam o mesmo tipo de dados. Isso significa que é muito mais fácil para duas (ou mais) linguagens inter-operarem, permitindo ao desenvolvedor escolher a linguagem de programação que melhor se aplica ao domínio da aplicação que está sendo desenvolvida.

Em resumo, o *Framework .NET* possui muitos recursos que o torna simples e produtivo, permitindo o desenvolvimento rápido de aplicações. Estes recursos incluem:

- integração de linguagens;

- neutralidade de plataforma;

- rica biblioteca de classes e serviços;

- gerenciamento automático de memória (*garbage collection*), etc.

Neste trabalho de pesquisa, iremos utilizar o *Framework .NET* para a construção dos serviços no nodo *gateway*. Assim, pretendemos construir uma interface Web e *Web Service* de forma a permitir que o usuário possa interagir de forma simples, eficiente e flexível (através de um PC, *Smartphone*, PDA ou telefone celular) com sua rede interna e privada através de uma rede pública como a Internet.

Para alcançar esse objetivo iremos estender as funcionalidades do *Framework .NET*e criar um novo *framework*, que utiliza os componentes do *Framework .NET,* mas agrega novas funcionalidades específicas para o domínio de aplicação em questão, tais como componentes para interagir com o nodo sorvedouro, componente para captura e codificação de vídeo, componentes para autenticação e autorização do usuário, etc.

Atualmente existem inúmeros *frameworks* para interação entre as aplicações disponíveis aos usuários finais e as redes de sensores [38]. Este trabalho visa também contribuir para a comunidade científica propondo um novo *framework* nesse nível.

A figura 13 ilustra o diagrama de blocos do *Framework .NET* e o local onde nosso *framework* a ser desenvolvido se encaixa.

Figura 13: *Framework .NET* + *Framework* para o Nodo *Gateway*

3.1.2 Plataforma Java

A plataforma Java é constituída de várias edições, tais como a *Java 2 Standard Edition (J2SE),* a *Java 2 Enterprise Edition (J2EE)* , a *Java 2 Micro Edition (J2ME)*, etc. Cada uma destas edições foi criada para atender a uma área específica do amplo campo da Computação. Abaixo, segue uma breve descrição de cada uma dessas edições:

- *Java 2 Platform, Enterprise Edition*: voltada para arquiteturas corporativas, que necessitam atender seus clientes, fornecedores e funcionários através de uma arquitetura de soluções servidoras escaláveis. O padrão J2EE é definido através de um conjunto de especificações relacionado; dentre estas especificações as principais são a EJB (*Enterprise Java Beans*), a especificação *Servlet* e a especificação JSP (*Java Server Pages*).

- *Java 2 Platform, Standard Edition*: voltada para o desenvolvimento de aplicativos *desktop,* para serem executadas em computadores clientes.

- *Java 2 Platform, Micro Edition*: projetado para atender as necessidades de desenvolver aplicações voltadas a dispositivos móveis, portáteis, que possuam limitações computacionais tais como memória, processamento, conectividade e diferentes sistemas operacionais.

Neste trabalho iremos utilizar o J2ME [39] para o desenvolvimento das aplicações que serão distribuídas em dispositivos tais como telefones celulares dos usuários. O desenvolvimento de aplicações para estes tipos de dispositivos é de fundamental importância, pois permitirá que os usuários interajam com sua rede privada a partir de dispositivos portáteis com acesso à Internet, independente de sua localização.

3.2 Padrões Arquiteturais, Modelos de Referência e Arquiteturas de Referência

Entre os esboços de diagramas que são considerados os pontos iniciais e embriões de uma arquitetura, até o estágio com todas as informações relativas ao sistema

levantadas, existem alguns passos intermediários de extrema importância. Cada um desses passos representa o resultado de um conjunto de decisões arquiteturais e as amarrações dessas escolhas e são muito úteis por si só. Em seguida são discutidos três dos passos intermediários mais importantes.

3.2.1 Padrões Arquiteturais

Um padrão arquitetural é uma descrição de um elemento e seus tipos de relações juntamente com um conjunto de restrições em relação a como esse elemento deve ser utilizado. Um padrão pode ser pensado como um conjunto de restrições em uma arquitetura – nos tipos de elementos e nos padrões de interação – sendo que estas restrições definem um conjunto ou família de arquiteturas que as satisfazem. Segundo Buschmann [40], um padrão descreve uma solução para um problema que ocorre com freqüência durante o desenvolvimento de software, podendo ser considerado como um par "problema-solução". Assim, pode-se pensar num padrão arquitetural como uma forma de identificar a essência de uma solução e torná-la suficientemente genérica para ser utilizada em problemas similares. O padrão funcionaria como um elo entre o problema e a solução. Os padrões arquiteturais seriam o nível mais alto de abstração de padrões para um projeto de sistema [40] sendo que a escolha do padrão arquitetural geralmente é o primeiro passo importante do arquiteto na projeção de um sistema. Por exemplo, um padrão arquitetural bastante comum é o padrão cliente-servidor. Cliente e servidor são dois tipos de elementos, e sua coordenação é descrita em termos do protocolo que o servidor utiliza para se comunicar com os clientes. O uso do termo "cliente-servidor" implica somente que múltiplos clientes existem, não são propriamente identificados e não há discussão sobre qual funcionalidade foi atribuída aos clientes e ao servidor. Inumeráveis arquiteturas diferentes podem seguir o padrão cliente-servidor sob o ponto de vista dessa descrição informal. Portanto, um padrão arquitetural não é uma arquitetura, mas ele exprime uma imagem útil do sistema, impondo restrições na arquitetura e, de fato, no sistema.

Um dos aspectos mais úteis nos padrões está no fato deles demonstrarem atributos de qualidade conhecidos. Isto justifica o fato dos arquitetos optarem por um padrão em particular e não um padrão aleatório. Alguns padrões representam soluções conhecidas para problemas de desempenho, outros endereçam melhor problemas de segurança e outros têm sido utilizados com sucesso para sistemas de alta-disponibilidade.

O custo de construção, evolução e manutenção de uma aplicação é fortemente influenciado pelas seguintes variáveis:

a) disponibilidade de pessoas qualificadas nas tecnologias e arquitetura da aplicação, e;

b) oferta de produtos, soluções e ferramentas que auxiliem no desenvolvimento da aplicação, em todo seu ciclo de vida.

Uma arquitetura estruturada ao redor de um conjunto de padrões arquiteturais e utilizando uma série de blocos de aplicação potencializam o entendimento da solução e, conseqüentemente, sua evolução por outros pesquisadores. Ao encapsular esses blocos de aplicação em componentes específicos temos uma melhor opção de evolução, não ficando susceptível a mudanças bruscas de padrões.

Embora um padrão defina uma solução para um domínio de problema específico, normalmente os sistemas estão envolvidos com diversos problemas de domínios diferentes, tornando impraticável a projeção de um sistema complexo baseando-se apenas em um único padrão arquitetural. O conjunto de diferentes requisitos faz com que os sistemas sejam projetados seguindo diversos padrões a fim de atingir o objetivo desse sistema.

Mesmo sistemas projetados seguindo a idéia de vários padrões arquiteturais, não há como garantir a qualidade da solução, pois a composição de uma nova arquitetura exige competência e estudo aprofundado para que se tenha a eficácia necessária a que se propõe. Além disso, mesmo arquiteturas consolidadas como a arquitetura de *Web Services*, podem apresentar falhas de concepção.

O termo “estilo arquitetural” também tem sido largamente utilizado para descrever os mesmos conceitos.

3.2.2 Modelos de Referência

Um modelo de referência é uma divisão de funcionalidade juntamente com um fluxo de dados entre as peças. Um modelo de referência é um padrão de decomposição de um problema conhecido em partes que, cooperativamente, resolvem este problema. Resultando de experiências práticas, modelos de referência são características dos domínios de maturidade e descrevem em termos gerais como as partes se inter-relacionam para alcançar seu propósito coletivo [41].

Alguns modelos de referência acabam sendo padronizados por Comitês de padronizações, os quais formalizam a decomposição de um problema conhecido. Um exemplo de modelo de referência bem consolidado é o modelo OSI da ISO [42]. Esse modelo define uma decomposição em camadas do problema de comunicação entre sistemas em um ambiente de rede. Outro modelo mais recente, mas bastante consolidado é o modelo de referência para Arquiteturas Orientadas a Serviços, da OASIS [43]. Organizações como BEA, IBM e Microsoft utilizam este modelo de referência como base para propor suas próprias Arquiteturas de Referência.

O uso de um modelo de referência é como um atalho para o planejamento de uma arquitetura de software, o qual permite que a elaboração de uma nova arquitetura não tenha que se preocupar com os elementos funcionais do domínio em questão, na medida em que estes elementos já foram identificados. Por exemplo, em uma arquitetura SOA o elemento responsável por prover conectividade entre diferentes tecnologias é o ESB (*Enterprise Service Bus*). Os elementos comuns de uma arquitetura SOA foram estudados e serão apresentados detalhadamente no Capítulo 5 desta tese, onde apresentamos uma Arquitetura Concreta para sistemas baseados em conceitos de Redes de Sensores e Atuadores Sem Fio, com foco para sistemas de atuação, monitoração e vigilância patrimonial.

3.2.3 Arquiteturas de Referência

Uma arquitetura de referência é um modelo de referência mapeado em elementos de software (que cooperativamente implementam a funcionalidade definida no modelo de referência) e os fluxos de dados entre eles. Considerando que um modelo de referência divide a funcionalidade, uma arquitetura de referência é o mapeamento dessa funcionalidade em uma decomposição do sistema [41]. O mapeamento pode ser, mas não necessariamente, um para um. Um elemento de software pode implementar parte de uma função ou muitas delas.

Uma Arquitetura de Referência constitui um conjunto de boas práticas e recomendações de arquitetura de sistema que visa padronizar o desenvolvimento e evolução de aplicações. Geralmente, uma Arquitetura de Referência apresenta um conjunto de definições arquiteturais independente de tecnologia e plataforma.

Uma Arquitetura de Referência tem como objetivos alavancar principalmente:

a) melhor custo benefício financeiro na construção de novas funcionalidades nas aplicações, tornando-as mais modulares e flexíveis;
b) minimização dos custos de manutenção das aplicações;
c) construção de aplicações "*Future proof*": mais fáceis e com menos custo financeiro de serem evoluídas;
d) alinhamento aos padrões .

Podem existir diversas arquiteturas de referência para um mesmo modelo. Isso ocorre porque há muitas maneiras de se transpor um modelo de referência em uma arquitetura de referência e, além disso, alguns modelos de referência são tão genéricos que podem ser aplicados em domínios de problemas diferentes. Pelo mesmo motivo, podem existir diferentes arquiteturas de software para uma arquitetura de software.

Atualmente, os três pilares de uma Arquitetura de Referência são:

- Arquitetura Orientada a Serviços (SOA, *Service Oriented Architecture*);
- Arquitetura orientada a componentes, e

- Arquitetura baseada em padrões.

3.3 Visão Geral dos Conceitos de uma Arquitetura Orientada a Serviços

O acrônimo "SOA" vem se tornando cada dia mais conhecido e contraditório. Duas pessoas poderiam definir SOA de maneira diferente e até mesmo de forma contraditória. Alguns poderiam descrever SOA como uma infraestrutura de Tecnologia da Informação para alavancar negócio enquanto outros poderiam visualizar em SOA oportunidades para aumentar a eficiência de TI. O desafio de definir SOA tem se tornado tão importante que vários consórcios de fornecedores de software e organizações de padronização vêm lançando iniciativas para tentar responder a questão "O que é SOA?".

Para o propósito deste trabalho, definiremos SOA como uma abordagem arquitetural onde um sistema é decomposto em partes menores (componentes ou serviços autônomos) que fornecem determinadas funcionalidades ao expor um número de serviços [44] e encoraja o reuso desses componentes. Esses serviços devem possuir baixo acoplamento e devem ser inter-operáveis entre si, podendo ser facilmente compartilhados dentro e/ou fora da organização.

Os conceitos arquiteturais relacionados a SOA não são novos – muitos deles evoluíram de idéias e experiências associadas a arquiteturas e desenvolvimento de sistemas distribuídos baseados em tecnologias disponíveis anteriormente. Muitos dos conceitos associados a SOA tais como serviços, descoberta e *late-binding* foram associados à CORBA e DCOM. Da mesma forma, muitos princípios de projeto de serviços já foram propagados pelo Paradigma de Orientação a Objetos (POO), como,por exemplo, a definição de interfaces, o baixo acoplamento, o encapsulamento e isolamento do cliente dos detalhes da implementação.

Diferentemente dessas iniciativas anteriores, a promessa chave de SOA é alavancar processos de negócio através de padrões abertos de interoperabilidade, tal como *Web Services*, por exemplo. SOA prevê integração no inicio da fase de planejamento de um sistema, ou seja, a solução final pode ser vista como uma composição de serviços

desenvolvidos em diferentes linguagens de programação, hospedados em plataformas diferentes com variados modelos de segurança e processos de negócio.

Em uma arquitetura SOA, os serviços devem ser combinados para implementar processos de negócio. As principais característica dos serviços são [43]:

a) interface bem definida, exposta para todos os clientes que necessitam do serviço (portal Web, aplicações internas, etc);

b) mecanismo de invocação do serviço, através da sua interface;

c) encapsulamento da implementação do serviço e desacoplamento desta implementação e seus clientes.

A Arquitetura Orientada a Serviços, no entanto, impulsiona estas boas práticas para um patamar superior, o da Arquitetura da aplicação. A implementação da abordagem proposta por este paradigma em todos os níveis de uma RSASF torna possível o rápido desenvolvimento e os testes de novas aplicações.

3.3.1 A evolução da Arquitetura Orientada a Serviços

Orientação a Serviços é a evolução natural dos modelos de desenvolvimento atuais. Na década de 80, viu-se surgir o modelo orientado a componentes; na década de 90 surgiu então o modelo de desenvolvimento orientado a componentes; agora temos o modelo orientado a serviços. Esse modelo mantém os benefícios do modelo de desenvolvimento orientado a componentes, mas há uma mudança significativa no paradigma: ao invés da chamada remota de métodos em componentes utiliza-se a passagem de mensagens entre os serviços. Os *schemas* descrevem não somente a estrutura das mensagens como também os contratos, os quais definem padrões para trocas de mensagens e políticas que definem a semântica dos serviços. Esta mudança promove a interoperabilidade, na medida em que as mensagens podem ser enviadas de um serviço a outro sem levar em consideração como a implementação do serviço trata estas mensagens. A figura 14 ilustra um serviço enviando e

recendo mensagens, as quais devem seguir os *schemas*e os contratos.

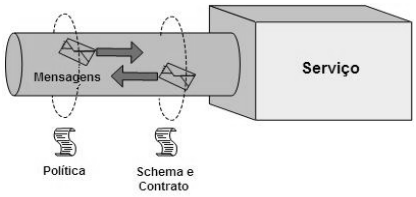

Figura 14: Anatomia de um serviço

Dessa forma, no contexto deste trabalho, uma mensagem pode ser enviada a partir de uma aplicação instalada em um dispositivo móvel (um *smartphone*, por exemplo) para o nodo *gateway* de uma RSASF e este por sua vez pode endereçar a mensagem para um nodo sorvedouro. Este último interpreta a mensagem e a encaminha para um serviço hospedado no nodo sensor ou atuador de destino da mensagem. Diferentes arquiteturas de hardware foram envolvidas no processo de comunicação do dispositivo móvel até o nodo sensor/atuador. Além dessa interoperabilidade entre dispositivos heterogêneos, a implementação de qualquer um dos serviços pode ser alterada sem que o cliente desse serviço necessite alterar a forma como se comunica com o serviço modificado, desde que não ocorra quebra no contrato (conceitos de integração fracamente acoplada e resistência a mudanças).

Embora muitas implementações freqüentemente utilizem padrões pré-estabalecidos como *Web Services*, orientação a serviços é independente de tecnologia. Assim, orientação a serviços é um conjunto de princípios arquiteturais expressados de forma independente de qualquer produto, assim como os conceitos de desenvolvimento, tais como polimorfismo e encapsulamento são independentes da linguagem de programação.

A unidade fundamental de uma arquitetura orientada a serviços é o serviço. Um serviço é um programa que pode interagir por meio de trocas de mensagens bem definidas. Serviços devem ser projetados para oferecer tanto disponibilidade como estabilidade. Um dos maiores benefícios de SOA é permitir agilidade no desenvolvimento de novas aplicações ou sistemas, a partir da infraestrutura já existente ou construindo aplicações compostas a partir do reuso de outros serviços – uma organização com processos de negócio implementados em uma infraestrutura fracamente acoplada é muito mais aberta a mudanças do que uma organização restrita por aplicações monolíticas que requerem semanas para implementar uma pequena mudança de software.

Em uma arquitetura SOA, os serviços e suas respectivas interfaces devem

permanecer estáveis, permitindo-se apenas que sejam re-configurados ou re-agregados para atender a uma necessidade de negócio.

3.4 Arquitetura Orientada a Componentes de Negócio

Os serviços disponibilizados são construídos e categorizados em função dos domínios de negócio aos quais eles se aplicam: os Componentes de Negócio do barramento de serviços.

Esta agregação dos serviços em Componentes de Negócio visa principalmente definir as fronteiras dos domínios de negócio: vários serviços que cooperam entre si no processo de negócio.

A organização dos elementos estruturais da arquitetura em componentes de negócio também facilita a documentação e publicação dos serviços disponíveis, através de soluções de gerenciamento de ativos digitais (governança de serviços), alavancando a reutilização destes componentes.

3.5 Arquitetura Baseada em Padrões

O custo de construção, evolução e manutenção de uma aplicação é fortemente influenciado pelas seguintes variáveis:

a) disponibilidade de pessoas qualificadas nas tecnologias e arquitetura da aplicação;

b) oferta de produtos, soluções e ferramentas que auxiliem no desenvolvimento da aplicação, em todo seu ciclo de vida.

Uma arquitetura estruturada ao redor de um conjunto de padrões arquiteturais e utilizando uma série de blocos de aplicação potencializam o entendimento da solução e, conseqüentemente, sua evolução por outros pesquisadores. Ao encapsular esses blocos de aplicação em componentes específicos temos uma melhor opção de evolução, não ficando susceptível a mudanças bruscas de padrões.

Um padrão arquitetural descreve onde ele pode ser aplicado e se é possível sua aplicação considerando outras restrições de projeto, as conseqüências e os custos do seu uso [45]. Também fornece dicas de implementação e exemplos. A solução é um arranjo de objetos e classes que solucionam o problema, sendo então adaptada e implementada para resolver uma questão num contexto em particular.

3.5.1 Visão Geral do Microsoft WCF (Windows Communication Foundation)

O *Windows Communication Foundation* é um conjunto de ferramentas para o desenvolvimento de software (*SDK – Software Development Kit*) que permite desenvolver serviços para a plataforma Windows. WCF provê um ambiente de execução para os serviços, permitindo expor tipos do CLR (*Common Language Runtime*) como serviços. O WCF é a implementação da Microsoft de um conjunto de padrões que definem interações entre serviços, conversões de tipos e mensagens e gerenciamento de vários protocolos de comunicação. Devido a isso, WCF provê interoperabilidade entre serviços além de prover funcionalidades tais como segurança, chamadas assíncronas, gerenciamento de transações, confiabilidade, gerenciamento da instância do serviço, etc.

No WCF, todas as mensagens trocadas entre o cliente e o serviço são mensagens SOAP. No entanto, as mensagens são independentes do protocolo de transporte – diferentemente de *Web Services*, os serviços WCF podem se comunicar através de inúmeros protocolos e não somente através do protocolo HTTP.

Devido ao fato de um serviço não possuir sua implementação divulgada aos seus clientes, um serviço WCF geralmente expõe metadados que descrevem as

funcionalidades disponíveis e as possíveis formas de comunicação com o serviço. Esses metadados são publicados de uma forma pré-definida e independente de tecnologia, tal como a forma WSDL sob HTTP-GET ou algum outro padrão para troca de metadados. Assim, um cliente não-WCF pode importar os metadados para o seu ambiente como tipos nativos. De forma similar, um cliente WCF pode importar os metadados de um serviço não-WCF e consumi-los como interfaces e classes nativas do CLR.

Com o WCF o cliente nunca interage com o serviço diretamente, mas sempre através de um *proxy*, que encaminha a chamada ao serviço. Os *proxies* expõe as mesmas operações que os serviços mais alguns métodos de gerenciamento do *proxy*. Em uma mesma máquina, o cliente pode consumir o serviço no mesmo domínio de aplicação, entre domínios de aplicação diferentes no mesmo processo ou então entre processos diferentes. A arquitetura dessa comunicação é ilustrada na figura 15. Entre computadores diferentes, o cliente pode interagir com o serviço através da rede local ou através da Internet, conforme ilustrado na figura 16.

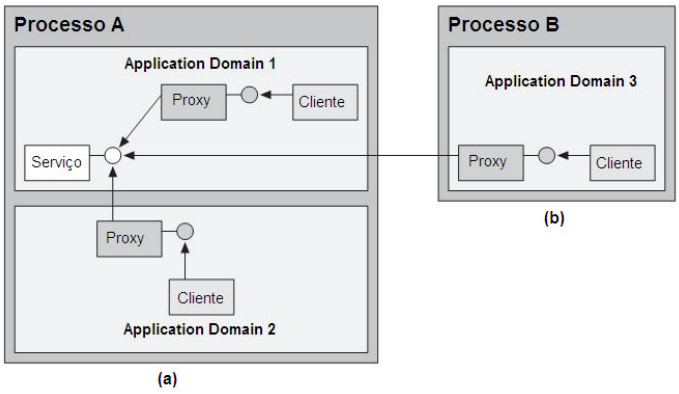

Figura 15: Arquitetura de Comunicação do WCF entre *Application Domains* (a) e entre processos diferentes (b)

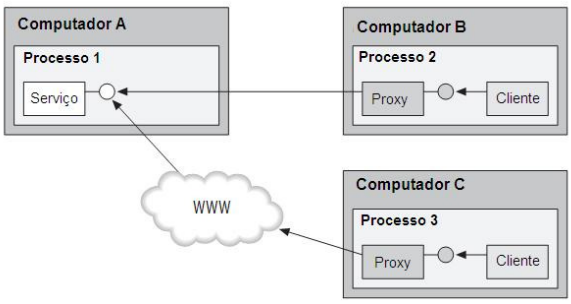

Figura 16: Arquitetura de Comunicação do WCF entre computadores diferentes

3.5.1.1 Endereçamento no WCF

No WCF todo serviço é associado a um único endereço. O endereço fornece dois elementos importantes: a localização do serviço e o protocolo de transporte utilizado para se comunicar com o serviço. O elemento de localização do serviço é composto pelo nome do servidor ou do domínio onde está hospedado o serviço, uma porta de comunicação e uma URI (*Universal Resource Identifier*, que pode ser um string que identifica um serviço único). Os protocolos suportados pelo WCF são: HTTP, TCP, Peer Network, MSMQ (Microsoft Message Queue) e IPC (comunicação inter-processo através de *named pipe*). Dessa forma, um endereço de um serviço sempre terá o seguinte formato:

[endereço base]/[URI]

[endereço base] = [protocolo de transporte]://[máquina ou domínio]:[porta]

3.5.1.2 Contratos WCF

No WCF os serviços são expostos através de contratos. O contrato é uma forma padronizada, independente de plataforma, e descreve a funcionalidade do serviço. WCF define quatro tipos de contratos:

a) contratos de serviços: descrevem quais operações o cliente pode realizar no serviço;
b) contratos de dados: definem quais os tipos de dados podem ser passados entre o cliente e o serviço;
c) contratos de falha: definem quais erros são levantados pelo serviço e como esse serviço gerencia tais erros e os propaga ao cliente;

d) contratos de Mensagem: permitem que os serviços interajam diretamente com mensagens. Contratos de mensagens podem ser tipados ou não tipados e são úteis em casos de interoperabilidade e quando existe um formato de mensagem que deve ser obedecido.

3.5.1.3 Hospedagem de serviços WCF

Todo serviço WCF deve ser hospedado em um processo Windows chamado processo hospedeiro. Um único processo hospedeiro pode hospedar inúmeros serviços e o mesmo serviço pode ser hospedado em vários processos hospedeiros.

3.5.1.4 Bindings

Dado um serviço, existem inúmeros aspectos de comunicação e padrões de comunicação possíveis: mensagens podem ser síncronas (na forma *request/reply*) ou assíncronas (*fire-and-forget*); mensagens podem ser bi-direcionais, podem ser entregues imediatamente ou armazenadas em mecanismos de fila; as filas podem ter suas mensagens persistidas em disco ou apenas na memória. Existem vários protocolos de comunicação para as mensagens, tais como HTTP (ou HTTPS), TCP, P2P, IPC ou MSMQ. Existem algumas opções de codificação: pode-se escolher por *plain-text* para permitir interoperabilidade entre plataformas diferentes, codificação binária para aumentar o desempenho ou MTOM (*Message Transport Optimization Mechanism*) para o caso de grandes cargas pagas (*payloads*). Existem também algumas opções para segurança de mensagens: pode-se optar por não protegê-las, fornecer segurança somente no nível de transporte, fornecer segurança e privacidade no nível da mensagem e, evidentemente, existem inúmeras formas de autenticar/autorizar os clientes. O serviço pode ter a necessidade de inter-operar com outros serviços ou clientes que são compatíveis apenas com o protocolo de *Web Services*, ou ter a necessidade de se comunicar com serviços que utilizam os benefícios do WS-*, tais como WS-Security, etc.

3.5.1.5 Endpoints

Todo serviço WCF é associado a uma terna de elementos. O primeiro elemento é o endereço que define onde o serviço está localizado, o segundo é o *binding* que define a(s) forma(s) de se comunicar com o serviço e, por ultimo, o contrato, que define qual a funcionalidade do serviço. Esta tupla que governa o serviço é muitas vezes referenciada como o "ABC" do serviço. O padrão WCF formaliza este relacionamento na forma de um *endpoint*. Assim, um *endpoint* é a fusão do endereço, do contrato e do *binding*do serviço.

Todo serviço WCF deve expor no mínimo um *endpoint*e cada *endpoint*possui exatamente um único contrato. Todos os *endpoints* de um serviço possuem o mesmo endereço, e um único serviço pode expor vários *endpoints*. Esses *endpoints* podem utilizar o mesmo ou diferentes *bindings* e podem expor um único ou diferentes contratos. Não há nenhum relacionamento entre os diferentes *endpoints*que um serviço pode expor.

3.5.1.6 Confiabilidade

O WCF e outras tecnologias orientadas a serviços fazem distinção entre confiabilidade de transporte e confiabilidade de mensagem. Confiabilidade de transporte (tal como aquela fornecida por protocolos de comunicação orientados à conexão, como o TCP) oferece garantia de entrega ponto-a-ponto na camada de rede assim como garante a ordem de envio/recebimento dos pacotes.

Já confiabilidade de mensagem, como o próprio nome sugere, trata da confiabilidade na camada/nível da mensagem independentemente de quantos pacotes são necessários para entregar essa mensagem. Confiabilidade de mensagem fornece garantia de entrega e da ordenação das mensagens, fim a fim, independente de quantos foram os intermediários envolvidos e de quantos foram os saltos necessários para entregar a mensagem do cliente para o serviço. Confiabilidade de mensagem é um padrão de comunicação baseado em mensagem confiável e que mantém sessão no nível de transporte. Esse padrão reenvia a

mensagem em caso de falha na camada de transporte; ele automaticamente controla o fluxo de dados, congestionamento, etc.

3.5.2 O padrão MVP para Camada de Apresentação

O padrão *Model View Presenter*, ou MVP, é suportado por meio da utilização do *Web Client Software Factory* (WCSF) [46]. O MVP é implementado pelo WCSF através do padrões *View, Presenter* e *Application Controller*.

O MVP é derivado do MVC [47] e tem o objetivo de suportar o modelo de aplicação do MVC e com foco na utilização de um observador dos três componentes do MVC. Ao invés de um *controller*, o padrão tem um *presenter*, mas a idéia permanece a mesma: o *view* apresenta os dados, o *model* armazena os dados e o *presenter* coordena a aplicação.

O *presenter* difere do *controller* porque tem mais poder de processamento, e tem o propósito de interpretar os eventos e executar qualquer lógica necessária para mapeá-los aos comandos que manipulam com o modelo (*model*), além de assumir as responsabilidades da aplicação sem um componente intermediário.

O padrão *Application Controller* define a criação de um objeto padrão para controle e centralização do fluxo das páginas e a lógica de navegação. O *view* interage com este objeto para executar o fluxo da página e a lógica de navegação da tela. Para isso, mantém e gerencia o estado entre as execuções do usuário. Também pode interagir com o *model* para executar ações de negócio e definir o fluxo da aplicação baseado no estado do modelo.

Com a utilização do WCSF é possível isolar a lógica da camada de apresentação e executar testes unitários sem utilizar a interface gráfica, melhorando a modularidade e manutenibilidade da aplicação.

3.5.3 O padrão Transfer Object (TO)

O padrão de projeto *Transfer Object*, ou objeto de transferência, é usado para mapear os objetos para conceitos do domínio do negócio. Estes objetos podem ser utilizados em todas as camadas, pois não contém lógica de negócio, somente dados que representam os atributos do conceito que fazem referência.

Em uma aplicação distribuída, estes objetos são serializáveis e diminuem o tráfego de rede, pois minimizam o número de *round-trips* que devem ser realizados entre o servidor *Web* e o servidor de aplicações, problema muito comum quando se utiliza *DataSets* ou outros objetos complexos. Desta forma, a utilização deste padrão visa também aumentar o desempenho da aplicação final.

Tanto a camada de serviços quanto a camada de negócio podem obter estes objetos a partir de objetos *DAO* e retorná-los para a camada de apresentação ou de orquestração. O caminho inverso também é possível, ou seja, um *Transfer Object* pode ser passado para as camadas de serviço, orquestração ou negócio, executam alguma lógica de negócio sobre ele ou utilizam um objeto *DAO* para persistir os dados do objeto.

3.5.4 O padrão Data Access Object (DAO)

Os objetos que implementam este padrão são responsáveis pela persistência e recuperação das entidades de negócio. Abstraem e encapsulam todo acesso à fonte de dados, deixando transparente para camada de serviços como os dados são obtidos e armazenados em meio persistente, independente de sua origem (banco de dados, arquivos *XML*, sistemas legados, etc.). Sendo assim, isola a lógica de acesso aos dados e os detalhes da conexão, facilitando sua manutenção, pois caso se necessite mudar a fonte de dados, isto não afetará as outras camadas da aplicação. Esta é a maior vantagem da utilização deste padrão.

A criação de uma instância de um tipo que implementa este padrão é feita através de um objeto que implementa o padrão *Factory* (ou fábrica). O padrão *factory* permite que a camada de acesso a dados seja desacoplada das camadas que fazem acesso a ela, minimizando o impacto no código cliente.

CAPÍTULO 4 – METODOLOGIA PARA CONSTRUÇÃO DA ARQUITETURA ORIENTADA A SERVIÇOS

Este capítulo tem o objetivo de fazer um mapeamento e levantamento realista dos principais serviços de uma RSASF e a metodologia para a construção de uma arquitetura orientada a serviços. Nesse mesmo capítulo também apresentamos o framework que foi desenvolvido para prover serviços genéricos de infraestrutura para a RSASF.

4.1 Metodologia para a construção de arquiteturas orientadas a serviços

Neste trabalho, a metodologia apresentada em [48] será utilizada para a construção de uma descrição arquitetural e o detalhamento da integração do usuário final com sua Rede Sem Fio. Os objetivos principais de cada etapa desta metodologia são apresentados em seguida.

- Etapa 1: Caracterização do Domínio:

Decompor o domínio do problema em cadeia de valor, considerando as áreas funcionais. Decompor cada área funcional em processos e sub-processos de negócio. Identificar os casos de uso de negócio, que são candidatos a serviços, pois oferecem uma visão das funcionalidades de negócio.

- Etapa 2: Criação de um Modelo de Serviços Objetivo:

Validar quão completa foi a identificação dos serviços feita anteriormente.

- Etapa 3: Análise dos Sub-Sistemas:

Refinar os casos de uso de negócio em casos de uso de sistema, tornando-os capazes de suportar os processos de negócios. Analisar o fluxo do processo de cada subsistema para identificar os candidatos a componentes de negócio. Utilizar os requisitos não-funcionais para encontrar os componentes tecnológicos. Identificar as funcionalidades necessárias para cada componente de negócio.

- Etapa 4: Alocação de Serviços:

Os serviços, previamente identificados, devem ser alocados de forma a corresponder aos objetivos do negócio. Definir em que componente o serviço será implementado e gerenciado.

- Etapa 5: Identificação dos Componentes:

Especificar cada componente de infraestrutura básica ou de negócio que está no escopo do projeto, apresentando suas regras, serviços, atributos (elementos de dados), dependências com outros componentes e pontos variáveis (regras, requisitos e *workflow* configuráveis).

- Etapa 6: Mapeamento para tecnologia de *Middleware:*

Definir os mecanismos para implementação dos componentes e serviços. Exemplos de *Middleware* orientados a objeto e serviços disponíveis atualmente: COM+, Corba, Java RMI, .NET Remoting, WCF, EJB (*Enterprise Java Beans*), etc.

4.2 Visão Geral do Estudo de Caso: Redes Sem Fio para Monitoração e Atuação

Nesta seção, consideramos as etapas da metodologia descrita na seção anterior para apresentar uma descrição arquitetural de um sistema construído a partir dos princípios de uma Rede de Sensores e Atuadores sem Fio para permitir que um usuário interaja com sua Rede sem Fio (por exemplo, acendendo ou apagando uma luz, ligando um dispositivo eletrônico, etc.) e receba eventos disparados por gatilhos pré-programados (como por exemplo, detecção de intrusão, abertura de uma janela ou porta, detecção de fumaça ou gás, etc.) através de qualquer dispositivo móvel ou PC conectado à Internet. Nesse ambiente, o usuário deve acessar o provedor de serviços, que faz o papel de nodo *gateway* da RSASF, de forma a ter acesso aos recursos de sua rede privada. Como o nodo *gateway* deve possuir funcionalidades que exigem mais recursos computacionais, resolvemos criar uma célula especial para alocá-lo, chamada de provedor de serviços.

O Estudo de Caso será abordado detalhadamente no Capítulo 6. Entretanto, faz-se necessário uma breve descrição neste momento para que possamos aplicar a metodologia descrita em [48] a fim de elaborar a descrição arquitetural baseada em algum domínio de aplicação. De todo modo, a arquitetura que será apresentada deve ser flexível suficiente para atender outros domínios de aplicação das RSASF.

4.2.1 Etapa 1 : Caracterização do Domínio

No estudo de caso deste trabalho foi projetado um sistema baseado nos conceitos de Redes de Sensores e Atuadores Sem Fio para prover atuação, monitoramento e segurança patrimonial. Através desse sistema um usuário pode interagir remotamente, através de uma interface amigável, com dispositivos de hardware complexos. Esta interação permite que o usuário do sistema capture dados do ambiente onde está instalada a RSASF (temperatura ou umidade do ambiente, por exemplo), capture em tempo real *stream* de vídeo (através das câmeras de monitoração ou vigilância), seja notificado (por e-mail, SMS, etc) da ocorrência de algum evento (detecção de intrusão, presença de determinado gás ou fumaça no perímetro monitorado, por exemplo) e atue modificando o estado de algum dispositivo (por

exemplo, acendendo ou apagando uma lâmpada elétrica, ligando ou desligando um sistema de irrigação, controlando a temperatura do ambiente, etc). Como na maioria das vezes o usuário do sistema pode encontrar-se em trânsito e precisa acessar o sistema remotamente, devem ser disponibilizadas algumas maneiras/interfaces de acesso, como por exemplo, através de um *smartphone* ou celular.

Como um sistema de RSASF envolve dispositivos complexos e heterogêneos, o desenvolvimento de aplicações se torna extremamente complexo, pois requer dos desenvolvedores conhecimentos de diferentes linguagens de programação, diferentes plataformas e arquiteturas de hardware, etc. e geralmente todo o sistema desenvolvido é exclusivo a um único domínio de aplicação. Essas questões tornam o processo de desenvolvimento de novas aplicações para RSASF lento e pouco produtivo.

Visando atenuar esses problemas, nesse estudo de caso segregamos o nodo *gateway* em uma célula especial, chamada de "Provedor de Serviços". Neste nodo especial, foi criada uma infraestrutura de componentes básicos hospedados como serviços e que foram projetados para atender todo e qualquer domínio de aplicações para as RSASF (como componentes de segurança, log/auditoria, cachê, sessão, etc) permitindo a reutilização desses serviços.

O nodo *gateway* é o intermediário entre o usuário final e os dispositivos sensores e atuadores, sendo o responsável por fazer a conversão dos protocolos utilizados e por fazer o roteamento da comunicação entre o usuário final e a rede interna (onde estão instalados o nodo sorvedouro e os nodos sensores e atuadores).

O domínio do problema pode ser decomposto em cadeia de valor, processo de negócio e casos de uso. A cadeia de valor, constituída de um conjunto de áreas funcionais é apresentada na figura 17.

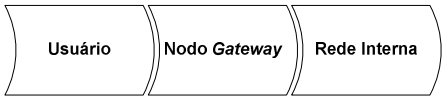

Figura 17: Cadeia de Valor do Domínio

Os domínios Usuário, Nodo Gateway e Rede Interna, podem ser descritos pelos casos de uso de negócio apresentados nas Tabelas 1, 2 e 3.

Usuário
UC100: Autenticar no Sistema
UC102: Consultar dados na Rede Interna
UC103: Modificar estado de dispositivos na Rede Interna
UC104: Assinar Publicação de *Stream* de Vídeo
UC105: Configurar o Sistema

Tabela 1: Casos de Uso para o domínio *Usuário*

Nodo *Gateway* / Provedor de Serviços
UC001: Autenticar Usuário
UC002: Rotear dados entre Usuário e a Rede Interna
UC003: Rotear dados entre a Rede Interna e o Usuário
UC004: Converter protocolos
UC005: Notificar usuário de evento na Rede Interna

Tabela 2: Casos de Uso para o domínio Nodo *Gateway*

Rede Interna
UC201: Rotear dados entre o Provedor de Serviços e os nodos sensores/atuadores
UC202: Rotear dados entre os nodos sensores/atuadores e o Provedor de Serviços
UC203: Converter protocolos
UC204: Atuar no ambiente
UC205: Capturar dados do ambiente
UC206: Notificar Usuário

Tabela 3: Casos de Uso para o domínio Rede Interna

4.2.2 Etapa 2 : Definição de um Modelo de Serviços Objetivo

O modelo de serviços objetivo para esse sistema deve:

a) fornecer flexibilidade ao usuário nas formas de acessar o sistema: Aplicação *Desktop,* Aplicação Web*,* Aplicação em *Smartphones*, Aplicação em celulares. O conjunto de funcionalidades disponíveis ao usuário decresce da primeira forma de acessar o sistema para última, em decorrência do poder computacional do hospedeiro do aplicativo cliente e das características das tecnologias envolvidas em cada tipo de aplicação;

b) fornecer informações on-line atualizadas;

c) fornecer possibilidade de reutilização e composição dos serviços básicos de infraestrutura instalados no Provedor de Serviços/nodo *gateway*, de forma que o novos sistemas, em diferentes domínios de aplicação, possam ser desenvolvidos com menores custos e investimentos financeiros;

d) fornecer ao desenvolvedor de serviços de negócio (serviços que estão diretamente relacionados ao domínio da aplicação) abstração da complexidade para integração com os componentes da Rede Interna, como por exemplo, o nodo sorvedouro e os nodos sensores e atuadores.

4.2.3 Etapas 3 e 4: Especificação dos subsistemas e alocação dos serviços

De forma a atender o modelo de serviços objetivo da seção 4.1.3, devemos ter alguns subsistemas fundamentais. Uma breve descrição desses subsistemas será explanada em seguida. Para melhor compreensão do relacionamento e da importância de cada subsistema, apresentamos na figura 18 um diagrama que mostra o mapeamento de fluxo e a visão funcional.

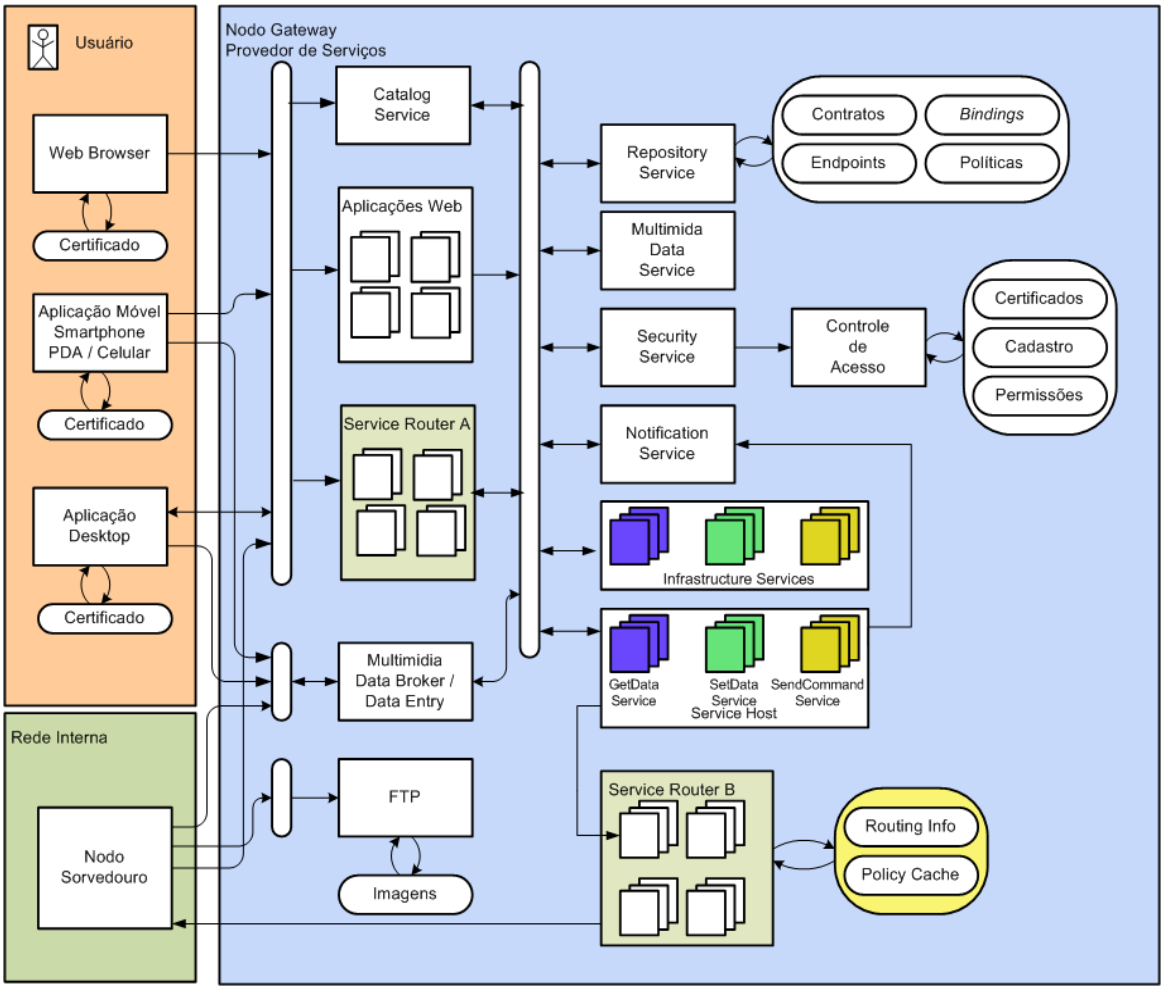

Figura 18: Visão funcional com o mapa de fluxo entre os subsistemas

O usuário pode acessar o sistema através de três interfaces: aplicação web; aplicação instalada em dispositivos móveis, como um *Smartphone*, PDA ou celular; ou através de uma aplicação *desktop*, instalada em um computador pessoal.

No primeiro caso, o usuário acessa o sistema através de aplicações Web, na forma *resquest/reply,* utilizando o protocolo HTTP ou HTTPS. Neste caso, toda a comunicação entre o nodo *gateway*e o usuário ocorre através do protocolo HTTP ou MMS, incluindo a difusão de dados multimídia. O sinal multimídia é convertido em vídeo e/ou som por um componente ActiveX instalado no navegador do usuário ou através da tecnologia *Silverlight,* que será apresentada no sexto capítulo (Estudo de Caso). Apenas na página em que o usuário envia seu *login* e senha para o sistema, é utilizado o protocolo HTTPS para proteger os dados enviados.

Nos segundos e terceiros casos, o usuário acessa o sistema através de *Web Services,* utilizando o protocolo HTTP para enviar comandos aos nodos atuadores, requisitar informações dos nodos sensores ou gerenciar o status de cada nodo. Para receber os dados multimídia, o cliente acessa o serviço *Multimidia Data Broker* (MDB), utilizando os protocolos HTTP, MMS ou TCP.

Já o nodo sorvedouro se comunica com o nodo g*ateway* através de um *socket* TCP para transmitir informações requisitadas pelo usuário e capturadas pelos nodos sensores, para receber comandos que deverão ser transmitidos aos nodos atuadores e para receber dados de gerenciamento de status dos nodos sensores/atuadores. O nodo sorvedouro envia os dados multimídia, capturados pelos nodos sensores (câmeras de vigilância/monitoramento), para o serviço *Multimidia Data Entry* (MDE) através do protocolo TCP.

O nodo sorvedouro se comunica também com um servidor de FTP, hospedado no nodo *gateway,* para enviar arquivos de imagem, capturados pelas câmeras de vigilância/monitoramento a partir do disparo de algum evento determinado pelo usuário (detecção de movimento na região monitorada, por exemplo).

A forma de comunicação entre usuário final e o nodo g*ateway* pode ocorrer de duas formas: Intermediada ou Direta. Na conexão Intermediada o consumidor do serviço requisita ponto de acesso ao serviço final para o subsistema *RouterService A*, que obtém o mesmo através de consulta ao subsistema *Service Registry.* O ponto de acesso exposto ao Usuário Final é provido pelo *Router Service A*, que mapeia este ponto de acesso com o ponto de acesso real do serviço presente internamente, conforme ilustrado pela figura 19. Já na forma de conexão Direta, o ponto de acesso exposto ao Usuário Final é o real, possibilitando assim a conexão direta com o serviço. A figura 20 ilustra este tipo de conexão.

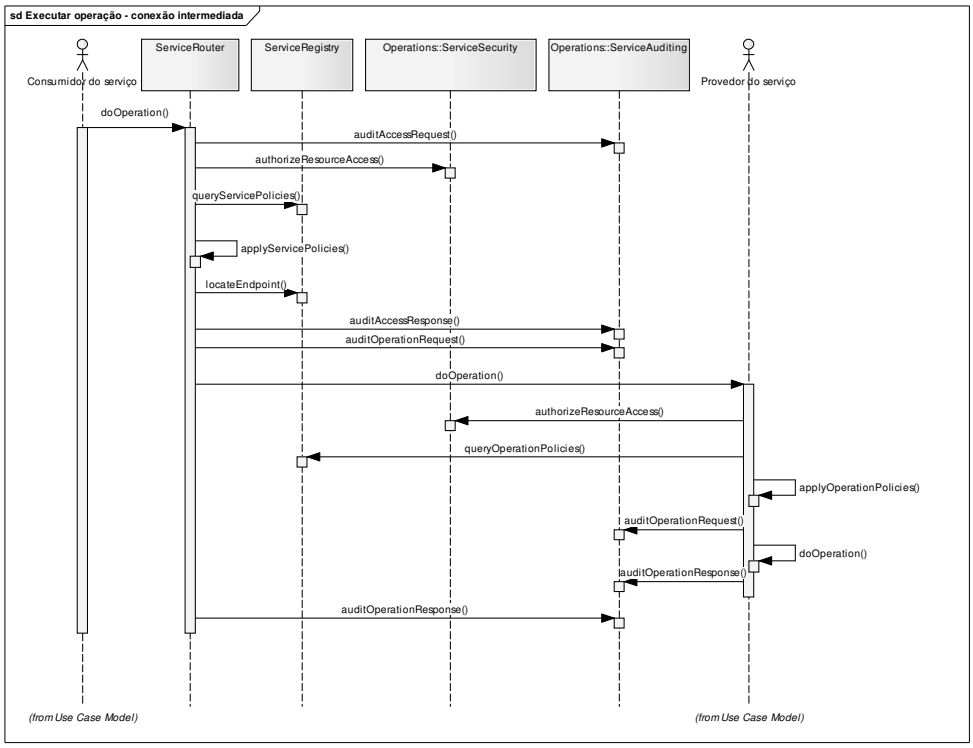

Figura 19: Diagrama de seqüência para o fluxo de executar operação através do serviço de roteamento

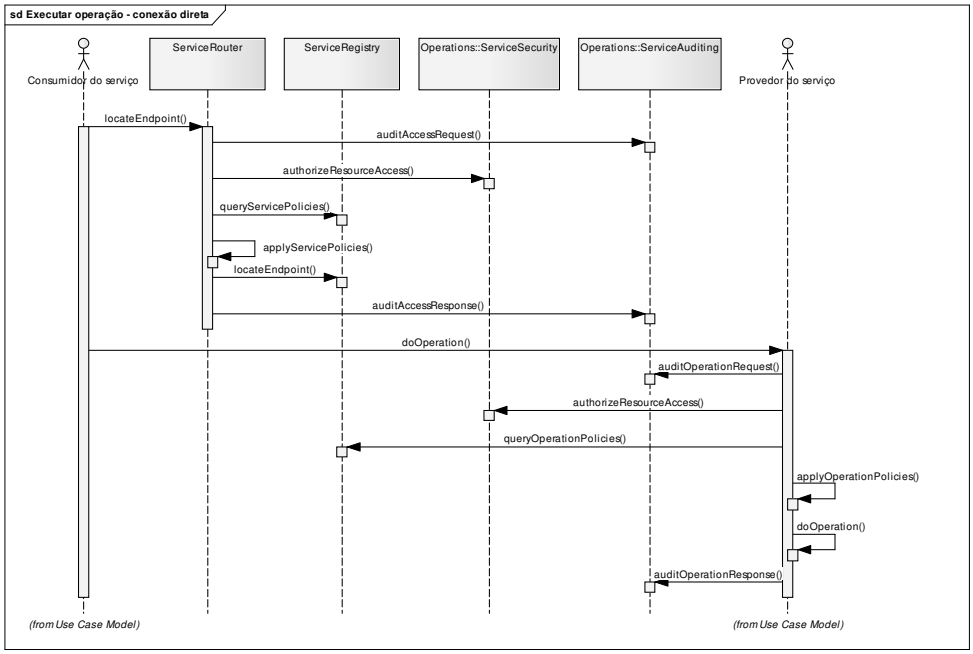

Figura 20: Diagrama de seqüência para o fluxo de executar operação sem o serviço de roteamento

Dentro do nodo g*ateway*, o primeiro subsistema, chamado de *Service Router A*, é responsável por intermediar a comunicação entre o usuário final e o nodo *gateway* para a maioria dos demais subsistemas. Esse subsistema deve fazer a conversão dos protocolos HTTP e HTTPS utilizados pelo Usuário Final para acessar os *Web Services* hospedados no nodo g*ateway,* para o protocolo TCP utilizado na comunicação com os serviços internos deste nodo. Intercepta as mensagens trocadas entre o usuário final e os provedores de serviços (com exceção da comunicação do usuário final com o subsistema *Multímidia Data Broker*), abstraindo assim os serviços dos seus pontos reais de acesso.

Já o subsistema *Service Router B* é utilizado na comunicação entre os serviços de negócio e o nodo sorvedouro. Tanto a comunicação dos serviços de negócio com o *Service Router B* e deste com o nodo sorvedouro são realizadas através de um *socket* TCP.

O subsistema *Multímida Data Broker* é responsável por difundir o sinal multimídia (como o *stream* de vídeo das câmeras instaladas na Rede Interna) para o usuário final. Esse sinal pode ser difundido através de vários protocolos, como por exemplo, HTTP e MMS, dependendo do dispositivo e da aplicação cliente que o usuário está utilizando para monitorar o ambiente da Rede Interna. O subsistema *Multimidia Data Entry (MDE)* é o ponto de entrada no nodo g*ateway* para o fluxo multimídia coletado pelas câmeras na Rede Interna e empacotado no nodo sorvedouro. O MDE apenas recebe o fluxo através do protocolo TCP e encaminha para o subsistema *Multimidia Data Service*, que por sua vez codifica o fluxo recebido para um determinado padrão multimídia.

O subsistema *Service Catalog*é o responsável por armazenar os metadados do serviço. Em sua implementação, consiste de três serviços:

- Um banco de dados no SQL Server 2005;

- Um serviço WCF de fachada que prove acesso aos metadados;

- Um serviço Windows chamado "*Microsoft MSE Repository Service*", que é o hospedeiro do serviço de fachada acima. Esse subsistema mantêm as informações necessárias para o consumo dos serviços e para aplicação das políticas para este consumo.

O subsistema *Service Host* é responsável por hospedar os componentes e serviços de negócio específicos do domínio de aplicação, como por exemplo, os serviços *GetData Service, SetData Service* e o *SendCommand Service*.

Já o *Infra Service Host* hospeda os componentes de infraestrutura, que são comuns para todos os domínios de aplicação. O *Infra Service Host* hospeda os componentes do arcabouço que foi desenvolvido para o nodo g*ateway* e que será apresentado na seção 4.1.5 deste capítulo.

Para divulgar a ocorrência de determinado evento capturado na Rede Interna e de interesse do usuário foi concebido o subsistema *Notification Service.* Ele recebe do nodo sorvedouro o sinal de que determinado evento ocorreu e verifica na base de dados qual a forma que o usuário deseja ser notificado para a ocorrência desse evento. Neste trabalho, foram identificadas duas formas de envio de notificação: por e-mail ou por SMS.

O subsistema *Security Service* é o módulo responsável por prover autenticação e autorização do usuário aos serviços disponibilizados. Ele intercepta todas as chamadas e verifica se o usuário está autenticado, se o *token* de autorização é válido e se o usuário está autorizado a realizar a operação requisitada. O diagrama de seqüência para o *Security Service* é ilustrado na figura a seguir. Para o caso das aplicações web, o *token* é armazenado em um *cookie* de sessão do navegador e o processo de autenticação e autorização é ilustrado no diagrama de seqüência da figura 21.

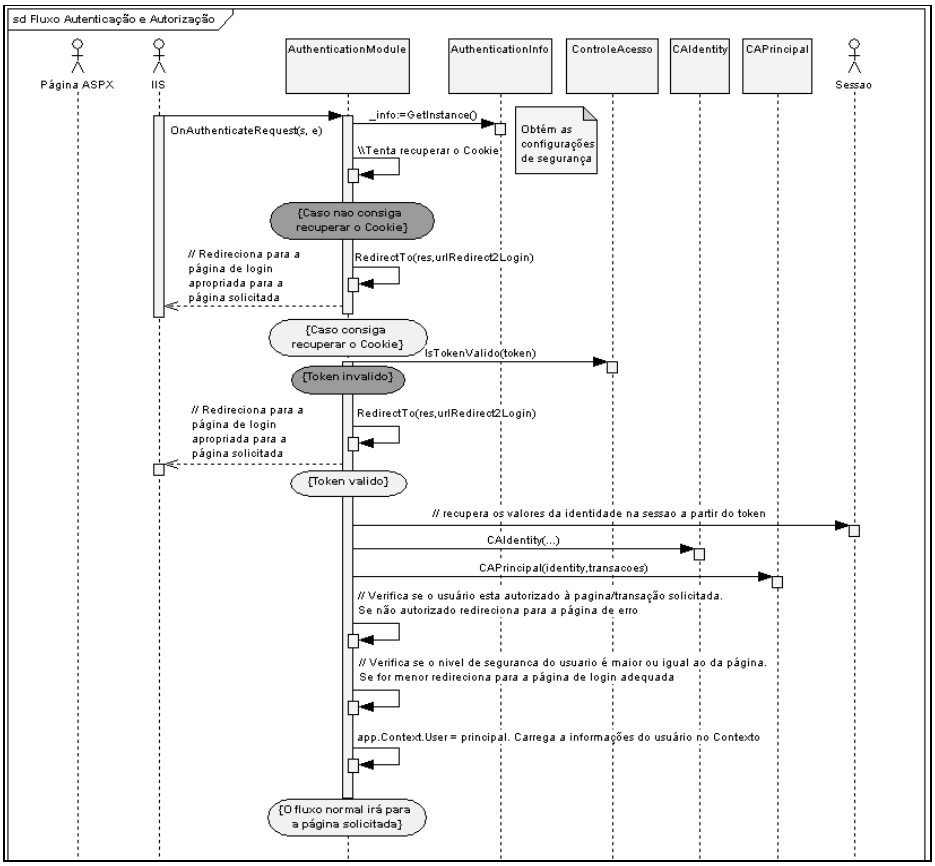

Figura 21: Diagrama de seqüência para o fluxo de autenticação e autorização do usuário no sistema

4.2.4 Etapas 5 e 6: Especificação dos componentes e mapeamento para a tecnologia de Middleware

Nesta etapa, tão importante quanto definir os componentes de negocio é identificar os componentes que são de uso comum em todos os domínios de aplicação de uma RSASF. O diagrama da figura 22 ilustra os componentes que foram projetados e desenvolvidos para constituir o arcabouço do nodo *gateway*. Os componentes foram segregados de acordo com suas funcionalidades: mecanismos de infraestrutura básica, de segurança e mecanismos de suporte para as camadas lógicas de Apresentação, Serviços e de Persistência.

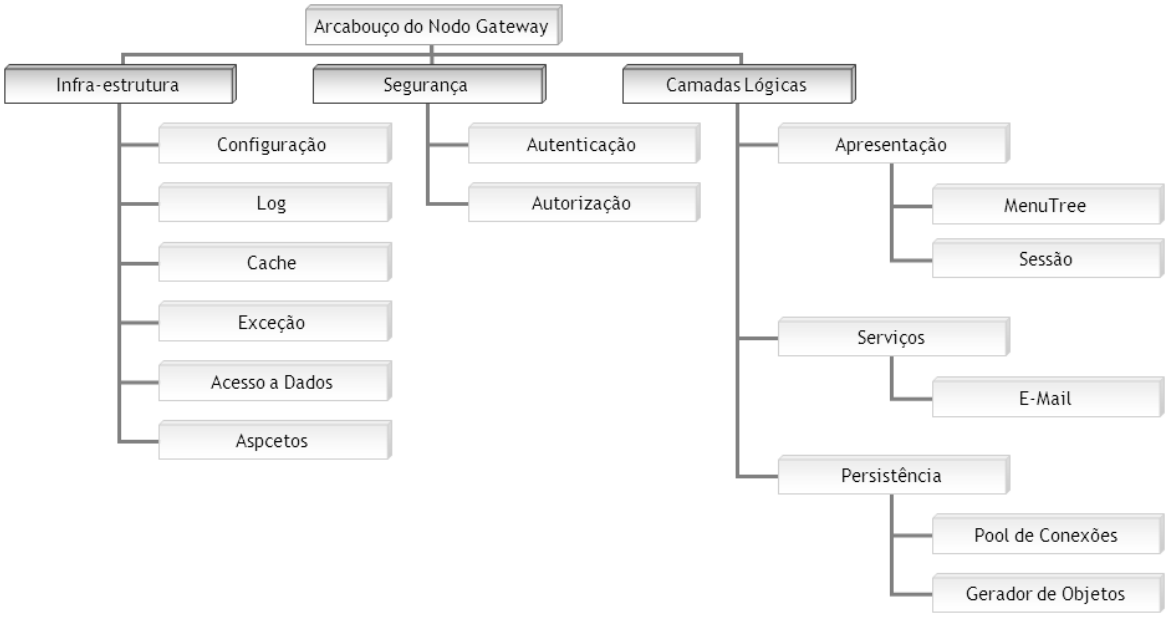

Figura 22: Principais componentes de infraestrutura do *framework* desenvolvido para o nodo *gateway*

Uma breve descrição de cada componente levantado é feito na seqüência.

- Mecanismo de Configuração: O mecanismo de configuração foi criado para manter toda configuração das aplicações centralizada em um único repositório, trazendo vantagens de manutenção e rastreabilidade. O mecanismo de configuração foi definido utilizando os seguintes princípios: o repositório deve ser único e centralizado; os dados devem ser mantidos no mecanismo de cachê para agilizar a consulta. Para a execução desse mecanismo temos um componente baseado no *Application Block* [49] de configuração disponibilizado pela Microsoft. A interface do componente com o desenvolvedor se dá através da classe estática *ConfigurationManager*.Através dela é possível recuperar informações que estão presentes na base de dados. Cada aplicação terá a configuração dos próprios componentes de infraestrutura (como log, sessão, acesso a dados, etc.), mais as configurações próprias do negócio. O mecanismo se utiliza do mesmo conceito do *.NET Framework* para configuração (equivalente ao que seria feito na configuração padrão em arquivo). Cria-se uma seção nova (um *XML*) com os dados da forma desejada. A partir desse *XML* deve-se criar um objeto para armazenar os dados. Em seguida cria-se uma classe *Handler* que sabe interpretar o *XML* e popular o objeto de dados. A configuração do componente de configuração é

bastante flexível. Ela é baseada na configuração do A*pplication Block*, modificado para ficar na base de dados.

- Mecanismo de Log: O mecanismo de log foi desenvolvido para permitir a identificação de erros de forma mais rápida e precisa durante a execução de determinado serviço ou aplicação. O mecanismo permite identificar se o erro é do serviço, se é um erro de negócio previsto no caso de uso ou se ocorreu devido a algum outro problema não previsto.

- Mecanismo de Cachê: O mecanismo de cachê foi desenvolvido para permitir o armazenamento de objetos de uso constante em memória para melhorar o desempenho da aplicação.

- Mecanismo de Exceção: O objetivo do mecanismo de exceção é identificar e gravar os erros ocorridos durante a execução de qualquer tipo de operação em um determinado serviço. O mecanismo de exceção tem dependência do mecanismo de log para realizar a publicação das exceções.

- Mecanismo de Segurança: O mecanismo de segurança, quando trata do aspecto de autenticação, visa garantir que quando um usuário tente entrar em uma aplicação, a permissão de acesso e a autenticidade da origem do *login* seja verificada. Determina quem está acessando a aplicação. Quando se trata do aspecto de autorização, o mecanismo visa garantir que um usuário não acesse informações ou execute operações confidenciais a ele, ou seja, determina o que o usuário pode acessar. O mecanismo de segurança realiza a validação do usuário no banco de dados de Controle de Acesso, cria o *token* de autenticação, retorna a lista de transações às quais o usuário está autorizado e as propriedades do usuário. Na autenticação, os dados do usuário são recuperados e se tornam acessíveis à aplicação. Esses dados podem ser recuperados no contexto da aplicação no objeto *Principal*.

- Mecanismo de Sessão: O *Framework .NET* possui três modos de controle de seção para aplicações Web: *inProc*, *outProc* utilizando base de dados SQL e *outProc*

utilizando o serviço *StateServer* do próprio *Framework .NET*. Para casos em que nenhum dos três mecanismos atende aos requisitos dos sistemas, foi criado um mecanismo próprio, tendo como requisitos aplicações distribuídas com balanceamento de carga, em que a camada de apresentação não pode acessar diretamente a base de dados. O mecanismo de sessão tem dependência com o mecanismo de configuração.

- Mecanismo de Envio de e-mail: Com esse componente é possível realizar o envio de mensagens de e-mail através de um servidor SMTP sendo sua configuração realizada por serviço.

4.2.5.1. Serviços de Repositório, Registro, Roteamento e Virtualização de Serviços

O primeiro requisito de uma Arquitetura Orientada a Serviços bem projetada é permitir que os serviços internos que compõe um sistema sejam potencialmente reutilizáveis e que possam ser acessados externamente. Isto reduz o tempo e esforço para criar novas aplicações ou estabelecer novos processos, já que é necessário menos código. Virtualização de Serviços é uma abordagem para disponibilizar e gerenciar serviços básicos de infraestrutura que provêem funcionalidades comuns a todos os outros serviços. Assim, os desenvolvedores podem focar em construir novas funcionalidades, as quais podem ser plugadas nesse barramento de infraestrutura.

Outro desafio que uma Arquitetura Orientada a Serviços deve endereçar é a gerencia de mudanças, através do versionamento de serviços. À medida que um serviço é desenvolvido, ele tende a sofrer alterações. Os serviços estão sujeitos a dois tipos de alterações: alterações na sua interface a alterações na sua implementação. O desafio está em prover uma maneira de acomodar as evoluções dos serviços e ao mesmo tempo minimizar os efeitos das mudanças para os consumidores deste serviço.

Prover segurança do sistema de acordo com as políticas de segurança previamente definidas também é outro grande desafio para uma arquitetura SOA. Por exemplo, em um cenário típico de SOA, podem-se ter inúmeros clientes de serviços e vários

Web Services, cada um desses com políticas de segurança diferentes. Se cada serviço for desenvolvido independentemente, por aplicação, os desenvolvedores necessariamente deveriam se preocupar não somente com a lógica do serviço como também nas complexas formas para garantir a segurança, resultando em uma redundância enorme de código de infraestrutura básica (nesse caso, segurança) que ficam passiveis de alterações caso a política de segurança se modifique.

A utilização da tecnologia MSE (*Managed Services Engine*), da Microsoft, é uma ótima abordagem para o estabelecimento de uma Arquitetura Orientada a Serviços, utilizando virtualização de serviços. O MSE foi construído sob o WCF (*Windows Communication Foundation*) e a plataforma Microsoft Windows, permitindo o desenvolvimento e a disponibilização de serviços de forma simples e rápida, com suporte a versionamento, abstração, gerenciamento, roteamento e aplicação de políticas de segurança aos serviços. O MSE é um componente intermediário que expõe as funcionalidades dos componentes através da virtualização de serviços, provendo uma arquitetura confiável e flexível para a construção de uma infraestrutura SOA.

O principal objetivo do MSE é fornecer um ambiente configurável em tempo de execução e prover tempos de resposta otimizados para processar as requisições aos serviços. Ao separar a lógica do serviço de seu comportamento, o MSE permite definir as condições de “como” os serviços são utilizados e “onde” eles são processados, fornecendo uma solução flexível e dinâmica, baseada em uma arquitetura fracamente acoplada com uma topologia coordenada e flexível, na medida em que pode ser configurada para atender aos requisitos de diferentes aplicações, com diferentes requisitos de negócio, nos mais variados domínios de aplicação.

Como visto no capítulo três, um serviço é uma entidade dinâmica que serve como fachada à frente de uma ou mais implementações. A arquitetura do MSE define essa entidade através de um serviço virtual representado por um contrato e constituído de vários componentes:

- Endpoints;

- Mensagens;

- Políticas;

- Contexto;

- Implementação.

A implementação do MSE disponibilizada pela Microsoft contém alguns componentes e ferramentas desenvolvidas no *Framework* .NET. Os dois principais componentes do MSE são o *Service Catalog* e o *MSE Runtime.*

O *Service Catalog*é o responsável por armazenar os metadados do serviço. Em sua implementação, consiste de três serviços:

- Um banco de dados no SQL Server 2005;

- Um serviço WCF de fachada que prove acesso aos metadados;

- Um serviço Windows chamado "*Microsoft MSE Repository Service*", que é o hospedeiro do serviço de fachada acima.

O serviço *MSE Runtime*é o coração do sistema; é ele quem prove o mecanismo de virtualização de serviço e em sua implementação física é composto de apenas um único módulo, um serviço Windows chamado "*Microsoft MSE Runtime*".

CAPÍTULO 5 – PROPOSTA DE ARQUITETURA ORIENTADA A SERVIÇOS PARA REDES DE SENSORES E ATUADORES SEM FIO

Os dois principais pontos fortes da arquitetura orientada a serviços estão na capacidade da tecnologia alinhar negócios com TI e dar flexibilidade, ou adaptabilidade, aos negócios. Porém, estes pontos não resultam automaticamente através da utilização de SOA, e só são obtidos através da definição da arquitetura de forma cuidadosa. Neste quinto capítulo, a arquitetura proposta é apresentada através de um conjunto de visões que juntas visam cobrir os principais aspectos técnicos relativos ao desenvolvimento e implantação do sistema em questão. O objetivo é capturar e formalizar as principais decisões tomadas com relação à arquitetura do sistema.

5.1 Arquitetura proposta para o Nodo Gateway

Neste trabalho, dentre os componentes de uma Rede de Sensores e Atuadores sem Fio, o componente que mais se destaca é o nodo *gateway*. É através desse nodo especial que o usuário final consegue acessar remotamente os serviços de sua rede privada, seja capturando dados de um sensor, gerenciando o status de um dispositivo ou interferindo no ambiente através dos nodos atuadores. Dessa forma, o nodo *gateway* deve fornecer serviços tanto para os usuários como para o nodo sorvedouro, instalado na rede interna.

Esses serviços devem ser instalados e configurados em uma infraestrutura e arquitetura que permita aos desenvolvedores de aplicações para RSASF rapidamente construir novos sistemas, reaproveitando os componentes e serviços que já existem, reaproveitando a infraestrutura e focando apenas nos componentes de negócio que o domínio da aplicação possa exigir. Além disso, o desenvolvimento de serviços para o nodo *gateway* é muito menos complexo do que desenvolver serviços para os nodos sensores, sendo que estes últimos requerem conhecimentos de linguagens de programação de baixo nível (geralmente *assembly*

e C) e conhecimento específico da plataforma de hardware, sendo necessário trabalhar com código altamente especializado para realizar uma determinada tarefa, como por exemplo, desenvolver rotina para controlar os servos-motores através de modulação PWM, capturar o *stream* de vídeo de uma câmera e transferí-lo para o nodo sorvedouro através de um enlace sem fio, etc.

A arquitetura proposta para o nodo *gateway* neste trabalho foi alcançada através de estudos de diversos padrões arquiteturais, padrões de mercado, levantamento bibliográfico de pesquisas relacionadas à área de Redes de Sensores e Atuadores sem Fio, Domótica, Arquiteturas Corporativas, Metodologias, Processos de Desenvolvimento, etc. Nas próximas seções, será feito o detalhamento das diversas visões que definem a arquitetura projetada para o nodo *gateway*, documentando de forma elucidativa as tecnologias envolvidas, o relacionamento entre camadas, etc.

5.1.1 Visão Geral

Esta seção tem por objetivo descrever a visão funcional da arquitetura orientada a serviços proposta para construção do nodo *gateway*. Para entendimento dos tópicos a seguir neste trabalho, faz-se necessária apresentação da Visão Funcional da arquitetura, cujo objetivo é identificar os principais elementos do modelo e, também, o papel desempenhado por cada um (ilustrado pela Figura 23).

Esta visão também serve de base para explicar as tecnologias envolvidas na montagem de um ambiente SOA. A visão também leva em consideração alguns elementos existentes no ambiente de sistemas da rede interna do usuário, como os nodos sensores e atuadores.

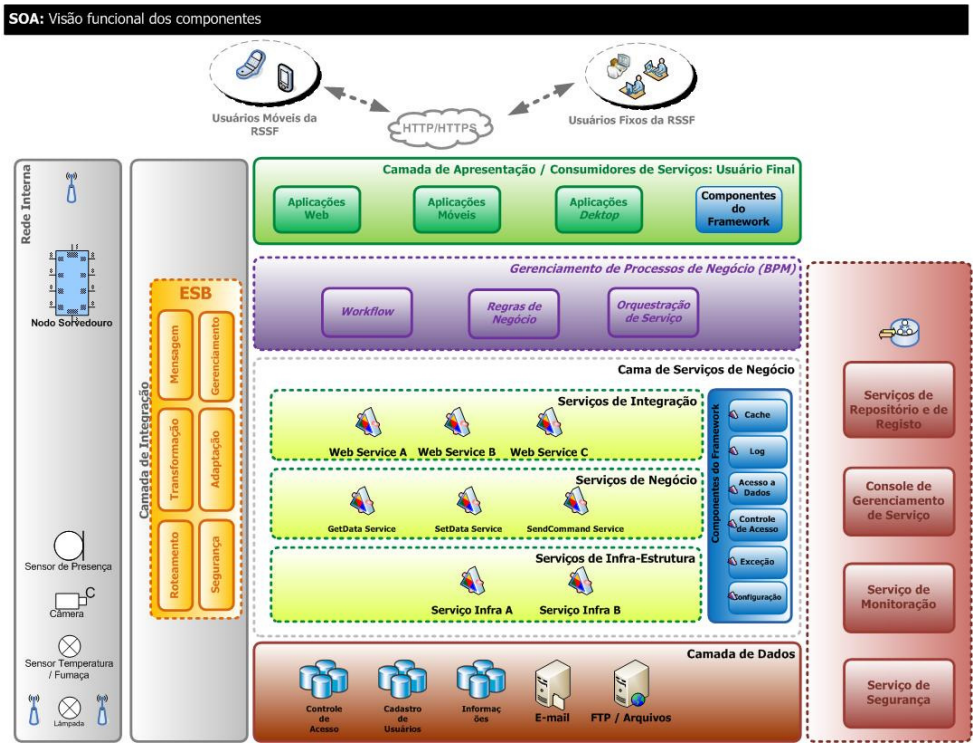

Figura 23: Visão Funcional da Arquitetura Orientada a Serviços

5.1.1.1 O Ambiente SOA

Para um suporte adequado a este ambiente é importante que a infraestrutura seja complementada com alguns componentes para dar suporte ao modelo de serviços. Esta infraestrutura oferece novas funcionalidades relacionadas a SOA e, pode ou não, interagir com a infraestrutura de aplicação existente. Este ambiente contém os seguintes componentes:

- Um ESB que fornece o mecanismo de processamento de mensagens e todo o suporte para o transporte, roteamento, segurança, adaptadores e transformação. Este componente oferecerá o suporte necessário para integração entre aplicações em diferentes plataformas ou através de diferentes protocolos. Este é o componente que interconecta os usuários finais, os quais podem utilizar aplicações clientes instaladas em dispositivos móveis ou em um PC, ou um navegador Web para interagir com a RSASF, através de protocolos como HTTP ou TCP. É através do ESB que o nodo sorvedouro se comunica com o

nodo *gateway*, enviando as solicitações dos usuários, informando a notificação de algum evento capturado pelos nodos sensores, enviando aos nodos atuadores os comandos emitidos pelos usuários, etc.

- Um serviço de registro e metadados (*Service Registry*) que é usado para localizar as informações sobre os serviços publicados no ambiente. Por exemplo, antes de consumir um novo serviço, tanto o analista de negócio quanto os desenvolvedores devem consultar se já existe algum serviço que ofereça o comportamento desejado, garantindo o re-uso e agilidade no processo de desenvolvimento.

- Um componente para publicação e descoberta de serviço (*Service Repository*) com suporte aos padrões do mercado, como por exemplo, UDDI (*Universal Description, Discovery and Integration*) [51]. O UDDI é um padrão específico para publicação e descoberta de serviços.

- Um mecanismo (*Business Rules Engine*) de execução e gerenciamento de regras de negócio.

- Um mecanismo de gerenciamento de serviço (*Service Management*) que permita controlar os vários aspectos dos serviços.

- Um mecanismo de segurança (*Security Service*) que permita controlar os vários aspectos de segurança dos serviços.

Os componentes listados acima são usados, principalmente, em tempo de execução, com exceção do componente de publicação e descoberta de serviços que também pode ser usado durante o desenvolvimento das aplicações.

Os componentes do modelo funcional, na sua maioria, usam o registro de serviços como repositório central para registrar e localizar todos os serviços e seus metadados.

O ESB, embora isolado na camada de integração, não significa que seja utilizado somente na comunicação com os componentes desta camada. Está agrupado na camada de integração devido ao papel que desempenha na arquitetura, porém também será usado para comunicação dos serviços e seus consumidores das outras camadas. Ideal para

cenários onde os consumidores foram construídos com tecnologias diferentes ou demandam alguma transformação de dados ou de protocolo.

No que diz respeito ao gerenciamento, este pode ser quebrado no nível de negócio e no nível de sistema. Para o nível de negócio, temos a Monitoração das atividades de negócio (BAM - *Business Activity Monitoring*). Para o nível de sistemas, temos: desempenho, ciclo de vida do serviço, configuração e mudança, controle de utilização e gerenciamento dos ativos.

O gerenciamento de identidade se refere à autenticação das identidades e o gerenciamento dos seus atributos, tais como papéis e privilégios. Como SOA permite um ambiente de maior colaboração entre organizações independentes, o gerenciamento de identidades é de extrema importância.

A segurança diz respeito à autenticação, autorização, integridade, confidencialidade, auditoria e privacidade.

5.1.1.2 Camada de Apresentação e Comunicação

Os usuários podem acessar os sistemas da RSASF através de diferentes canais. Por exemplo, os usuários podem acessar o nodo *gateway*através de suaIntranet, caso estejam conectados na mesma rede privada ou através da Internet, para acessarem a RSASF remotamente. Cabe lembrar que nem sempre são os usuários que interagem com os sistemas corporativos, podendo também ser outra aplicação que utiliza os serviços disponíveis para troca de informações, atuando como consumidores desses serviços.

Na figura 23, esta camada representa as várias formas de usuários e outros sistemas interagirem com as aplicações ou serviços corporativos. Como por exemplo, portais, aplicações web, aplicações *desktop* ou aplicações para dispositivos móveis.

5.1.1.3 Camada de Interface de Serviços

Na figura 23, esta camada tem o propósito de expor como serviço as funcionalidades e processos de negócio. Tais serviços devem ser independentes da camada de aplicação que eles abstraem, do ESB e da infraestrutura. Por exemplo, seja um serviço "*SendCommand*" desenvolvido para enviar comandos aos nodos atuadores da RSASF. Se a aplicação de envio de comandos aos nodos atuadores for substituída por outra, o serviço deverá produzir o mesmo resultado. Os consumidores dos serviços de negócio são independentes dos detalhes de implementação, ou seja, são fracamente acoplados dos produtores. Esta camada contém processos de negócio que são formados por outros serviços de negócio.

A construção dos objetos e classes que suportarão os serviços faz uso do "*Nodo Gateway Framework*", o qual representa os componentes existentes e encapsulam funcionalidades básicas e comuns a todos os serviços e aplicações.

As tecnologias utilizadas para construção desta camada devem oferecer suporte para definição e montagem dos processos de negócio, controle de acesso e segurança, execução de processos com suporte a transação, e comunicação através de protocolos padronizados.

Os serviços dessa camada podem ser classificados em:

- Serviços de Negócio: modularizam uma funcionalidade de negócio que é independente da aplicação que encapsula e está alinhado com os principais processos de negócio que uma RSASF deve prover.

- Serviços de componente (aplicação): são blocos de serviços que podem ser invocados diretamente, mas foram construídos com a intenção de servir como partes para composição de serviços de negócio que são montados de acordo com uma funcionalidade de negócio de alto nível.

- Serviço de dados: oferecem uma visão unificada das várias fontes de dados do nodo *gateway,* objetivando a abstração de recursos lógicos ou físicos.

- Serviços de infraestrutura: não estão no nível de negócio e podem ser utilizados entre as várias unidades de negócio, ou seja, não oferecem valor ao negócio diretamente.

Os serviços de componente são aplicações expostas como serviços de funcionalidade de negócio. Normalmente, estes serviços alavancam as aplicações que encapsulam e são desenvolvidos a partir de uma ou mais funcionalidades da aplicação. As tecnologias para construir esta camada devem oferecer suporte para controle de acesso e segurança, comunicação síncrona e assíncrona, transações e adaptadores.

Serviço de aplicação, serviço de integração e serviço utilitário são tipos especializados de serviço de componente que possuem as seguintes características:

- Mecanismo para integração entre aplicações customizadas, de terceiros ou aplicações legadas;
- Normalmente, não são centrados em negócio, e sim em integrações ponto a ponto, tais como transformação, tradução e evento;
- Pouca abstração de negócio e são projetados e desenvolvidos para uma solução específica ou que tenha valor direto ou indireto para o negócio ao invés de uma funcionalidade de negócio de alto nível. Realizam operações simples e não dependem de outros serviços.

Os serviços de dados são expostos para realização de operações de incluir, alterar, excluir e consultar informações de negócio, ou seja, entidades do modelo de negócio. Abstraem o repositório de dados e as complexidades de conexão para os consumidores destes serviços. As tecnologias para construir esta camada devem oferecer suporte para controle de acesso e segurança, comunicação síncrona e assíncrona e transações. Tem por objetivo abstrair recursos de dados lógicos e físicos e possuem as seguintes características:

- Abstraem as operações para incluir, alterar, excluir e consultar, e as instruções SQL equivalentes;
- Abstraem os recursos e fontes de dados através de interfaces;
- Abstraem a forma como os dados são persistidos;
- Garantem o modelo conceitual e a integridade dos dados.

Por último, os serviços de infraestrutura são usados para automatizar as tarefas de provisionamento de recursos para habilitação de um ambiente computacional compartilhado e virtualizado. Estes serviços possuem as seguintes características:

- Valor de negócio indireto;
- Não contém lógica de negócio;
- Provisão de recursos e ambientes;
- Encapsula detalhes da plataforma.

Reutilizam serviços de infraestrutura que limitam a necessidade de redundância de identidade e funções de segurança. Implementam serviços centralizados que eliminam redundância de diretório, reduz a complexidade de desenvolvimento e reduz o custo de operação.

5.1.1.4 Camada de Operação e Gerenciamento

Esta camada disponibiliza a funcionalidade que monitora e gerencia os serviços através de todas as camadas descritas anteriormente. Consiste de uma infraestrutura de descoberta de serviços que garante que os consumidores encontrem os detalhes, incluindo a localização de novos serviços e serviços atualizados.

A infraestrutura de segurança gerencia autenticação, controle de acesso e segurança dos dados para todas as camadas no modelo SOA.

O gerenciamento dos processos de negócio controla e gerencia o estado dos processos, transações de longa duração, correlação, entre outros.

O monitoramento e funcionamento do serviço são críticos para garantir os níveis de serviço.

As tecnologias usadas para construir esta camada devem oferecer suporte para definir, rastrear, gerenciar e monitorar os objetivos dos níveis de operação para todos os serviços, gerenciamento de processos de negócio (BPM) e capacidade para monitorar eventos de processo em tempo quase real, também chamado de *Near Time.*

5.1.2 Objetivos e Restrições

5.1.2.1 Objetivos da Arquitetura Proposta

A arquitetura proposta neste documento tem como principais objetivos: *flexibilidade*, *reusabilidade*, *disponibilidade*, *padronização de interfaces e modelo de dados*, *segurança, agilidade e foco no domínio de aplicação*, *composabilidade* e *interoperabilidade*.

A *flexibilidade* é obtida através da utilização de mecanismos de regras de negócio e automação de processos usando *workflows* que consomem serviços. Desta forma é possível fazer mudanças de forma rápida e com menor esforço, garantido um ambiente de TI capaz de se adequar à dinâmica e agilidade dos negócios. A utilização de um mecanismo de regras de negócio permite que uma aplicação seja alterada sem a necessidade de passar por um ciclo de desenvolvimento, dando flexibilidade e agilidade para atender as mudanças do negócio.

A *reusabilidade* permite que os recursos existentes de TI sejam encapsulados e modernizados através do encapsulamento por serviços, e que novas funcionalidades sejam disponibilizadas através da composição de serviços existentes. Outras aplicações podem reutilizar estes serviços sem se preocupar com os detalhes de construção e complexidade de acesso dos recursos por trás de cada serviço, além de reduzir o tempo de entrega de novas funcionalidades.

Visto que os serviços suportam protocolos padronizados e baseados em XML, por meio da solução de gerenciamento e monitoramento do ambiente SOA, e com infraestrutura de hardware adequada, é possível garantir um ambiente de alta disponibilidade.

A padronização das interfaces dos serviços e do modelo de dados reduz o custo de desenvolvimento de novas funcionalidades em ambientes de software heterogêneos e com dados incompatíveis. Os consumidores dos serviços não se preocupam com os detalhes de construção dos serviços e nem com o modelo de dados. Quando os dados disponibilizados pelo serviço não são compatíveis por um dos consumidores, o mecanismo de ESB oferece os

recursos necessários para resolver os problemas de integração e troca de mensagens entre aplicações distintas.

A *composabilidade* permite que novos serviços sejam construídos a partir de serviços existentes. Fornece mais agilidade para entrega ou alteração de uma funcionalidade e reduz a complexidade do ambiente de desenvolvimento. Sempre que for preciso desenvolver uma nova funcionalidade, faz-se uma consulta ao repositório de serviços para validar se já existe algum serviço que pode ser reaproveitado e, caso não exista, será desenvolvido um novo serviço. A composabilidade também pode ser obtida através da automação de processos usando *workflows* que consomem os serviços de negócio. Se houver alguma alteração no negócio que impacte o processo, somente o *workflow* será ajustado.

A *interoperabilidade* é obtida através da utilização de XML e *Web Services*, bem como dos padrões XML existentes, tais como WSDL (*Web Service Description Language*), UDDI, XML Schema, SOAP (*Simple Object Access Protocol)* e WS-*, e que são suportados por grande parte das linguagens de programação e plataformas de desenvolvimento.

5.1.2.2 Restrições

A arquitetura propõe a modelagem de processos de negócio através da tecnologia *Windows Workflow Foundation*[52], do Framework .NET 3.5 (WF). O WF utiliza o padrão XAML (*Extensible Application Markup Language*) [53], que é a linguagem utilizada pelo Framework .NET 3.5 para descrição de interface gráfica. Este mesmo padrão também é utilizado para descrição de *workflows* para WF, o que não tem suporte direto para o padrão BPEL (*Business Process Execution Language*) [54].

Como o BPEL é um dos principais padrões do mercado, é importante que a arquitetura tenha algum suporte para este padrão. A falta de suporte do WF ao padrão BPEL é um problema apresentado por vários usuários da tecnologia e para atender esta demanda, a Microsoft liberou um adicional chamado *BPEL for Windows Workflow Foundation*. Este adicional possui atividades para BPEL e uma ferramenta para importar e exportar de XAML

para BPEL, e vice versa. A versão suporta o padrão BPEL 1.1 e o padrão WS-BPEL (Web Services Business Process Execution Language) 2.0 [54].

Os serviços serão definidos, desenvolvidos e executados com o WCF, que também depende da plataforma Windows para ser executado. A dependência para executar os serviços na plataforma Windows não afeta os consumidores dos serviços em outras plataformas, até porque o WCF suporta os principais padrões XML do mercado.

5.1.3 Padrões Arquiteturais

Esta seção apresenta os padrões arquiteturais, tecnológicos e de infraestrutura que são propostas na arquitetura do nodo *gateway*.

5.1.3.1 Arquitetura Orientada a Serviços

A arquitetura tem como padrão a orientação a serviços e segue alguns princípios básicos, entre os quais se destacam:

- As funcionalidades de negócio são descritas, disponibilizadas e usadas como serviços;
- Os serviços são independentes da implementação;
- Os serviços podem ser combinamos para criação de novos serviços;
- Os serviços são descritos usando metadados;
- A descrição do serviço tem que ser significativa para os seus consumidores;
- A interação com os serviços é fracamente acoplada;
- Os serviços são publicados e descobertos num repositório, ou diretório, de serviços;
- Os serviços são invocados através de protocolos padronizados.

A Figura 24 ilustra uma visão tecnológica e o relacionamento dos componentes apresentados na visão funcional da arquitetura mostrada na seção 5.1.1 - Visão Geral. A visão técnica mostra quais tecnologias foram sugeridas para desempenhar as funções dos elementos essenciais da arquitetura, conforme a seção 5.1.1.1 - O ambiente SOA.

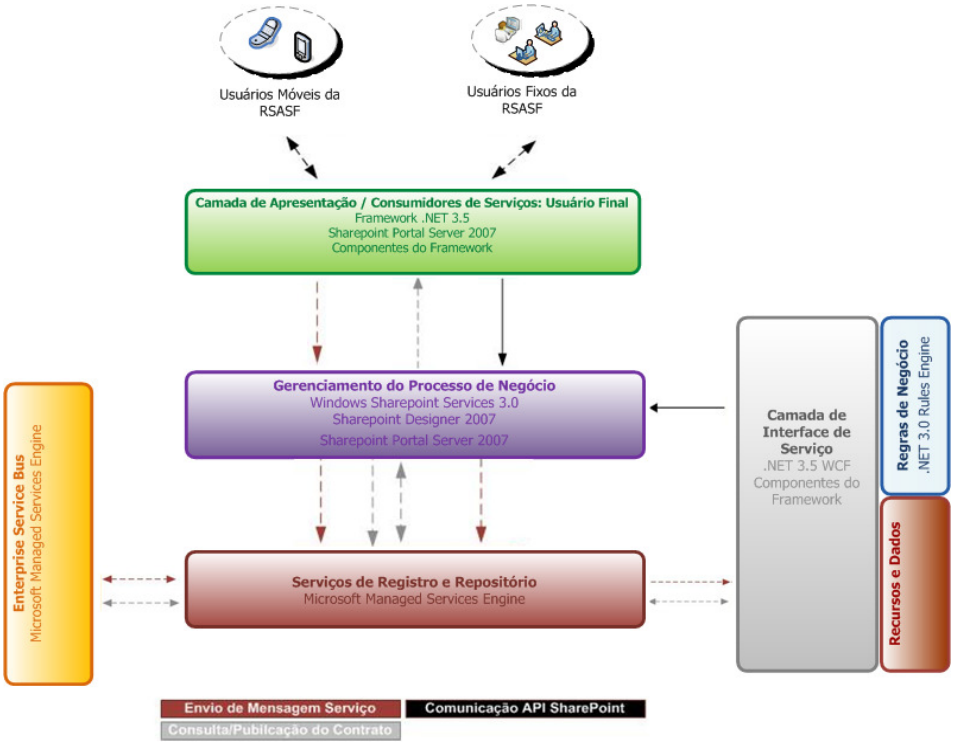

Figura 24: Visão tecnológica da arquitetura SOA e o relacionamento entre os componentes

5.1.3.2 Arquitetura Baseada em Padrões XML

Para garantir a interoperabilidade com plataformas e tecnologias distintas, a arquitetura adota tecnologias que suportam os principais padrões para XML e *Web Services*.

Como a arquitetura é predominada pela tecnologia Microsoft, pode-se concluir que a mesma suporta o padrão WS-Basic Profile 1.0 (BP) [55] do WS-I (*Web Services Interoperability Organization*) [56], visto que o padrão é suportado pela plataforma .NET. O WS-I é um consórcio formado pelos principais fornecedores de tecnologia (tais como, Microsoft, BEA, SAP, IBM) e tem por objetivo fornecer orientações de interoperabilidade para as principais especificações de *Web Services* para mensagens, descrição de serviços, publicação e descoberta de serviços. O WS-BP 1.0 suporta o XML (*Extensible Markup*

Language) 1.0 [57], SOAP (*Simple Object Access Protocol*) 1.1[58], WSDL (*Web Services Description Language*) 1.1[59], HTTP 1.1[60], XML Schema 1.0[61], e UDDI 2.03[62]/2.04[63].

5.1.3.3 Análise e Design Orientados a Serviços

A análise e design dos serviços levam em consideração princípios da orientação a serviços, tais como autonomia e reusabilidade. Além disso, a identificação dos serviços é feita com foco nos processos de negócio seguindo um modelo *top-down*, permitindo que os recursos de TI sejam utilizados da melhor forma possível e que agreguem valor ao negócio.

O modelo *top-down* assume que cada processo de negócio, ou função de negócio, é um candidato a serviço. Depois de identificados os serviços candidatos, um conjunto de passos bem definidos é executado de forma a garantir que todos os detalhes dos serviços sejam capturados, modelados e projetados para garantir um ambiente favorável ao reuso e composição.

5.1.3.4 Arquitetura Baseada em "Software Factories"

O desenvolvimento baseado em *Software Factories,* ou fábrica de software, soluciona um dos principais problemas do ambiente de desenvolvimento, onde aplicações são desenvolvidas e implantadas sem tirar proveito do conhecimento obtido com os artefatos produzidos na construção de outras aplicações similares. Algumas abordagens – treinamento, documentação e *frameworks* – são usadas para solucionar este problema, porém não permitem que o conhecimento seja aproveitado de forma eficiente e, também, são candidatos a erros de falha no processo.

O uso de fábricas de software soluciona este problema através da utilização de práticas para desenvolvimento de um tipo específico de aplicação dentro de um pacote integrado de guias de fácil adoção pelo time do projeto. O desenvolvimento de aplicações utilizando uma fábrica de softwareadequada proporciona vários benefícios:

- Consistência. Permite construir várias instâncias de um determinado software de linha de produção (que compartilha características e arquitetura), fazendo com que seja fácil a obtenção de consistência. Também simplifica a governança e diminui os custos de manutenção e treinamento.
- Qualidade. Facilita o aprendizado dos desenvolvedores e a implementação de práticas já testadas. Sendo assim, evita a perda de tempo com codificação desnecessária e faz com que o desenvolvedor mantenha o foco nas funcionalidades específicas da aplicação.
- Produtividade. Padroniza e automatiza as atividades no desenvolvimento da aplicação, através da: reutilização de artefatos (arquitetura, *frameworks*, blocos de aplicação); fornece guias contextualizados e automatizados; e geração de código a partir de modelos que representam abstrações dos mecanismos e elementos da aplicação.

A fábrica de software descreve uma coleção completa de recursos que podem ser usados para desenvolver aplicativos com uma tecnologia específica. Por exemplo, cada fábrica contém:

- Blocos e bibliotecas de aplicativos;
- Pacotes de receitas e guias como plug-ins que funcionam integrados à ferramenta de desenvolvimento;
-*Templates* para desenvolvimento dos componentes da aplicação;
- Implementação de referência (aplicação de exemplo que mostra como utilizar a fábrica) ; e
- Tópicos instrutivos.

A utilização destas *fábricas* garante produtividade e padronização para o ambiente de desenvolvimento de software. Como trabalham de forma integrada com a ferramenta de desenvolvimento, o desenvolver terá todo suporte necessário para executar suas

atividades de forma rápida e com qualidade conforme os padrões de desenvolvimento elencados. Além disso, também oferecem mecanismos que permitem que sejam customizadas de acordo com a necessidade.

As fábricas de softwares que foram estudadas e que serão adotadas como padrão da arquitetura proposta neste trabalho são:*Web Client Software Factory* (WCSF), *Web Service Software Factory*(WSSF)[64],*Smart Client Software Factory (*SCSF)[65] e *Mobile Client Software Factory* (MCSF)[66].

5.1.3.5 Arquitetura Baseada em Padrões de Projeto

A arquitetura proposta baseia-se nos principais padrões de projeto para construção dos componentes nas suas respectivas camadas. Entre os padrões utilizados, destacam-se: *Model View Presenter*, *Data Access Object*, *Transfer Object eService LocatorFactory,* apresentados no capítulo 3.

5.1.4 Visão Lógica

Esta seção apresenta a visão lógica da arquitetura proposta. Esta visão provê uma perspectiva estrutural do sistema, apresentando como este se divide em camadas, os principais elementos de design destas camadas, suas interfaces e relacionamentos.

5.1.4.1 Visão de Camadas

Esta seção descreve como o sistema está dividido em camadas, mostrando quais são elas, suas dependências e como se comunicam. A Figura 25 ilustra as camadas lógicas da arquitetura e seus relacionamentos.

Figura 25: Visão das camadas da arquitetura e seus relacionamentos

Cada camada desempenha um papel específico na arquitetura e, com base na figura 25, possuem as seguintes responsabilidades:

- Camada de Apresentação - é responsável por disponibilizar a interface gráfica por onde os usuários interagem com os sistemas. Oferece os recursos necessários para construção e controle da interface gráfica. Pode interagir diretamente com a camada de orquestração ou com a camada de serviços, além das entidades de negócio, ou objetos de transferência (*transfer objects*) da camada de objeto de negócio.

- Camada de Processo de Negócio - é responsável pela automação dos processos de negócios através de *workflows*. Os processos automatizados nesta camada podem fazer uso direto da camada de negócio ou da camada de acesso a dados.

- Camada de interface de serviço - é responsável por disponibilizar a interface, o contrato e a implementação dos serviços, seja serviço de negócio, de componente ou de dados. A descrição do serviço é formada pelo contrato, pela interface, pelos tipos de dados e tipos das mensagens, e pelo próprio serviço. Esta camada pode ser consumida diretamente pela camada de apresentação ou pela camada de processo de negócio. Pode se comunicar diretamente com as camadas de dados ou negócio.

- Camada de negócio - é responsável por disponibilizar os objetos que encapsulam a lógica de negócio e os objetos que mapeiam os conceitos do domínio do negócio. Esta camada não faz acesso direto ao banco de dados e

depende da camada de acesso a dados para qualquer comunicação com o banco de dados.

- Camada de acesso a dados - é responsável por encapsular a lógica para persistência e recuperação das entidades de negócio, abstraindo o mecanismo de persistência. Os objetos desta camada implementam o padrão de projeto DAO (*Data Access Object*).

- Camada de integração - é responsável pelos processos de integração entre sistemas, através de mecanismo de mensagens. Oferece um modelo desacoplado de integração e permite que diferentes sistemas conversem independente de protocolo ou formato de dados.

5.1.4.2 Camadas Lógicas: Tecnologias

Esta seção tem por objetivo descrever os principais elementos de projeto das camadas da arquitetura, bem como uma visão geral das tecnologias utilizadas para sua construção.

A figura 26 ilustra os principais componentes de design das camadas e como se relacionam. A figura separa os objetos de transferência em mais uma camada com o objetivo de ressaltar a importância destes objetos e ilustra que eles podem ser referenciados em qualquer camada.

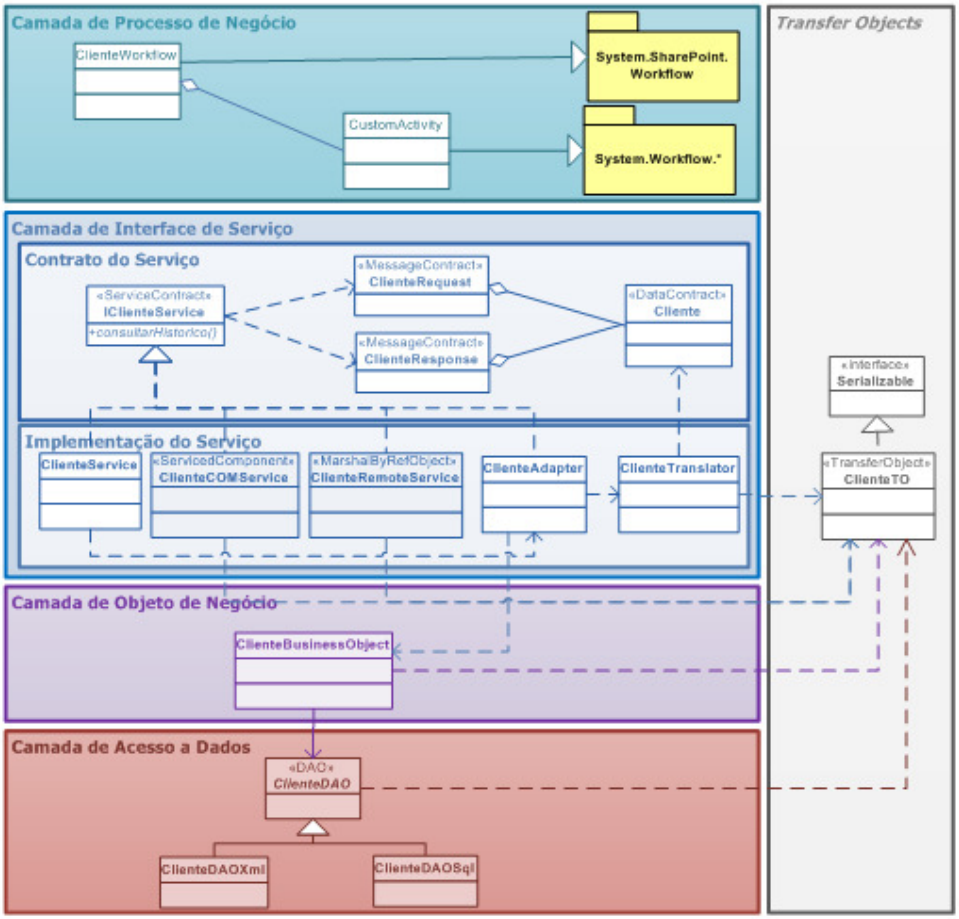

Figura 26: Visão detalhada das camadas da arquitetura

A camada de apresentação disponibiliza a interface gráfica do usuário através de componentes que determinam a fronteira do sistema. Permite abstrair a complexidade envolvida na interação com o usuário e a representação interna do sistema, além de gerenciar os elementos gráficos independente das regras de negócio da solução. A interface gráfica da aplicação é implementada com a tecnologia ASP.NET usando o *Web Client Software Factory* (WCSF).

A camada de apresentação poderá acessar qualquer uma das camadas inferiores, com exceção da camada de dados, que não pode ser acessada diretamente. É importante que a mesma acesse somente as camadas de orquestração, serviços ou negócio.

O mecanismo utilizado para suportar a construção de aplicações modularizadas e com total isolamento da lógica da interface e fluxo das páginas é baseado no padrão de projeto *Model-View-Presenter* (MVP), formado por três elementos básicos: *Model* (modelo),

View (visão) e *Presenter* (apresentador), padrão suportado pelo *Web Client Software Factory*. A figura 27 ilustra os principais componentes do padrão e os seus relacionamentos.

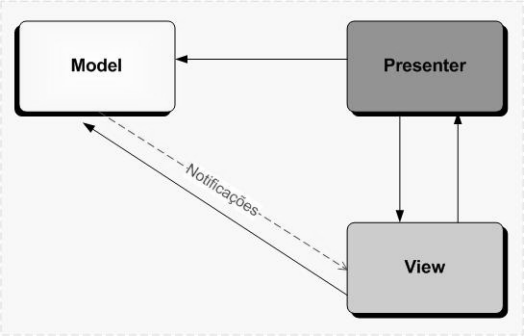

Figura 27: Representação do padrão de projeto MVC

O *Model* é o dado sobre o qual a interface do usuário irá operar. Representa um objeto do domínio e o objetivo é de que estes objetos não tenham conhecimento da interface do usuário. É puramente um objeto do domínio e em momento algum será usado pela interface.

O *View* tem a responsabilidade de mostrar o conteúdo de um *model*. É esperado que o *model* notifique o view sobre prossíveis mudanças nos seus dados, suportado pelo padrão de projeto *Observer*. É possível que mais de uma visão seja conectada a um único *model*. O view espera manipular os eventos da interface gerados pelo sistema operacional. Por exemplo, os dados de uma tela são tratados no evento do botão dentro do view, que em seguida repassa a chamada para um método de um *presenter*.

Enquanto que a responsabilidade do *view* é mostrar o modelo de dados, é o *presenter* que governa como o modelo pode ser manipulado e alterado pela interface do usuário. Ele está no núcleo dos comportamentos de uma aplicação, e a maioria do código, que trata como a interface do usuário trabalha, está dentro dele. A principal diferença é que o *presenter* é diretamente ligado a sua visão, onde os dois colaboram para prover a interface gráfica de um modelo em particular.

A camada de processo de negócio representa os artefatos de software criados para automatização de *workflows*. Estes *workflows* são classes de software que herdam de classes básicas do *Framework* .NET 3.5, e que têm seus fluxos modelados através de componentes (atividades) que desempenham um passo específico do processo. A arquitetura

propõe que estas classes também sejam consumidoras dos serviços de negócio. Assim, o processo de automação de uma determinada tarefa na RSASF deverá ser executado pelos *workflows* definidos.

Uma característica importante de uma arquitetura SOA é minimizar as dependências que cada serviço tem dentro do seu próprio processo de negócio. Serviços que contêm regras de negócio demandam que as regras sejam aplicadas em tempo de execução. Isto limita a capacidade do serviço de ser utilizado fora do ambiente que demande estas regras. Da mesma forma, serviços que atuam como controladores e embutem a lógica que compõe outros serviços podem gerar dependência da estrutura de composição.

A introdução de uma camada de controle acima de uma camada de serviços permite estabelecer uma localização central para regras de negócio e lógica de composição relacionada à seqüência em que as atividades são executadas.

A camada de processos de negócio é projetada especificamente para este propósito, introduzindo o conceito de "Serviço de Processo". O serviço de processo é capaz de compor outros serviços para realizar um processo de negócio de acordo com um *workflow* pré-definido.

Esta camada é suportada pelo *Workflow Foundation* e *WF Rules Engine* para descrição de *workflows* e execução de regras de negócio, respectivamente. Por padrão, a linguagem de descrição de *workflows* do WF é o XAML.

O mecanismo padrão para execução e controle de *workflows* é o *Windows SharePoint Services 3.0*, visto que dispõe dos recursos necessários para publicação, instanciação, execução, persistência e gerenciamento dos *workflows*. Além de oferecer um modelo de objetos que permite que estas funcionalidades sejam acessadas via programação.

A camada de serviços de interface introduz serviços projetados para representar a lógica de negócio, também chamado de Serviços de Negócio. São responsáveis por expressar a lógica de negócio através da orientação a serviços e de trazer a representação do modelo de negócio corporativo para o mundo dos *Web Services*.

Esta camada pode ser dividida logicamente em duas partes: *Service Contract*, definição do contrato do serviço (tipos de dados, tipos de mensagens, operações, esquemas XML), independente de tecnologia; e *Service Implementation*, implementação do serviço com uma tecnologia específica.

Os serviços são modelados como interfaces que declaram as operações disponibilizadas pelo serviço. A classe do serviço representa a implementação da interface, e contém a lógica de processamento de cada operação do serviço. Neste trabalho, tanto a definição da interface quanto a implementação do serviço são realizados com a tecnologia *Framework* .NET 3.5.

A comunicação com o serviço é feita através do seu Proxy, que nada mais é que uma classe que abstrai os detalhes de comunicação com o serviço, fazendo com que a execução das operações do serviço seja feita da mesma forma que a chamada de um método qualquer de uma classe. A vantagem da utilização do Visual Studio .NET 2005 como ferramenta para construção dos serviços é que o mesmo oferece recursos para geração automática do código do proxy. Para isso, só precisa que seja informado o endereço do arquivo de descrição do serviço, o arquivo WSDL.

A lógica de negócio implementada por um serviço pode estar numa classe de negócio específica ou dentro da própria classe do serviço. A vantagem de colocá-la fora da classe que implementa o serviço é que a mesma pode se reutilizada em outra aplicação, por exemplo. De qualquer forma, o fato de colocar a lógica de negócio dentro do próprio serviço não impede que o mesmo seja reutilizado, pelo contrário, pois a idéia é que este serviço seja consumido por qualquer aplicação que necessite desta lógica de negócio.

Outro ponto importante da utilização do *Framework* .NET 3.5 para camada de serviços está no suporte às principais extensões *Web Services* (WS-*), tais como WS-Transaction, WS-Security, WS-Policy, entre outros. Então, pode-se dizer que os serviços terão suporte a controle transacional distribuído, segurança declarativa, e distribuição da lógica de negócio (serviços distribuídos), bastando configurá-los de acordo com a necessidade.

Antes da codificação dos serviços, é importante que seja construído o diagrama do modelo de serviços. Este diagrama permite que os serviços sejam descritos sem se preocupar com os detalhes da tecnologia utilizada para implementação. Após a modelagem do serviço, configuram-se os detalhes da tecnologia de implementação e, em seguida, é gerado o código contendo todas as interfaces, mensagens, tipos de dados, tipos de falhas, entre outros. A lógica da classe que implementa o serviço deve ser implementada em outra classe que herda da classe gerada pelo WSSF. Isto permite que o código seja re-gerado sem que seja perdida a codificação feita anteriormente.

O *Windows Communication Foundation* oferece todo mecanismo necessário para o design e codificação do contrato dos serviços, bem como dos tipos de dados e tipos de mensagens. Além disso, permite que o serviço seja hospedado por qualquer aplicação .NET. Quando um serviço em WCF é executado, os clientes podem consumi-lo através da geração de uma classe *Proxy*.

Um serviço WCF é projetado de forma independente do canal de comunicação e do formato das mensagens, visto que são detalhes que podem ser especificados via XML no arquivo de configuração da aplicação que o hospeda.

Na figura 26, por exemplo, o contrato do serviço (*IClienteService*) faz referência as classes *ClienteRequest* e *ClienteResponse*, ambas decoradas com o atributo *MessageContract*. Estes atributos são específicos do WCF e permitem customizar como será o corpo da mensagem SOAP de entrada e de saída para as operações deste serviço. As classes, por sua vez, fazem referência a classe Cliente que também está decorada com outro atributo do WCF, o *DataContract*. Este atributo indica que uma classe é um tipo de dado utilizado por um serviço, seja como parâmetro de uma operação ou como campo de uma mensagem. A utilização dos atributos WCF para decorar as classes é uma forma de especificar como será o arquivo WSDL do serviço, ou seja, a descrição dos tipos de dados, dos tipos de mensagem e das operações.

Além dos tipos e operações, a descrição do serviço deve especificar, entre outras informações, qual o canal de comunicação do serviço e o endereço onde será executado. Outros detalhes do contrato do serviço são descritos através do arquivo de configuração da aplicação que hospeda o serviço, seja uma aplicação Windows ou ASP.NET.

A camada de acesso a dados é responsável pela persistência e recuperação das entidades de negócio. Os objetos desta camada suportam o padrão de projeto DAO. Abstrai e encapsula o acesso a fonte de dados, deixando transparente à camada de serviços como os dados são obtidos e armazenados em meio persistente, independente de sua origem (banco de dados, arquivos XML, sistemas legados etc.).

A principal vantagem de utilizar este padrão é que toda lógica de acesso aos dados fica isolada, facilitando sua manutenção, pois caso seja necessário mudar a fonte de dados não impactará as outras camadas da aplicação.

A instanciação de um objeto DAO é feita através de uma fábrica que desacopla as camadas e permite que sejam alteradas sem impactar o código do cliente. A figura 26 ilustra o relacionamento entre as classes da camada de persistência e entidades (TO's), e afigura 28ilustrauma das formas de ligação entre a camada de serviços e a camada de dados.

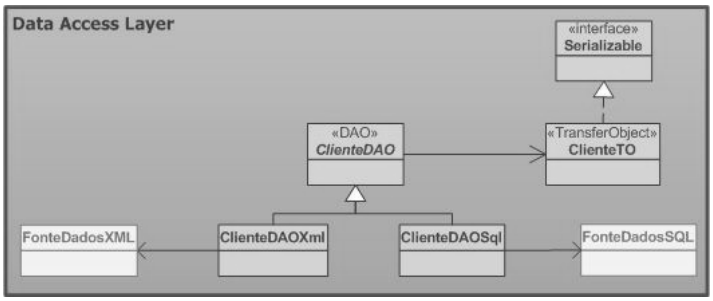

Figura 28: Design dos elementos da camada de acesso a dados

Para um determinado mecanismo de persistência de dados e para cada entidade persistente da aplicação, três classes são criadas e colaboram entre si. No diagrama de seqüência da figura 29, onde é representada uma entidade genérica (Entidade), estas classes são: Entidade, EntidadeDao e EntidadeDaoImpl.

Outras implementações do DAO também podem ser criadas para outras fontes de dados. Por exemplo, no diagrama, a classe EntidadeDaoXmlImpl estende o DAO para obter os dados de arquivos XML. Utilizando um exemplo concreto, se em um sistema é necessário a obtenção dos dados da entidade Usuário, deve-se criar as classes Usuário, UsuárioDao e UsuárioDaoImpl.

As responsabilidades das classes desta camada são as seguintes:

- EntidadeDao - Classe abstrata que define os métodos de persistência utilizados pela camada de serviços. Podem ser definidos métodos para criação, atualização, remoção ou obtenção de dados. Estes métodos sempre recebem ou retornam *Transfer Objects* correspondentes à entidade.
- EntidadeDaoImpl - Estende a classe abstrata, implementando os métodos de persistência definidos, para uma determinada tecnologia de persistência (RDBMS como padrão).

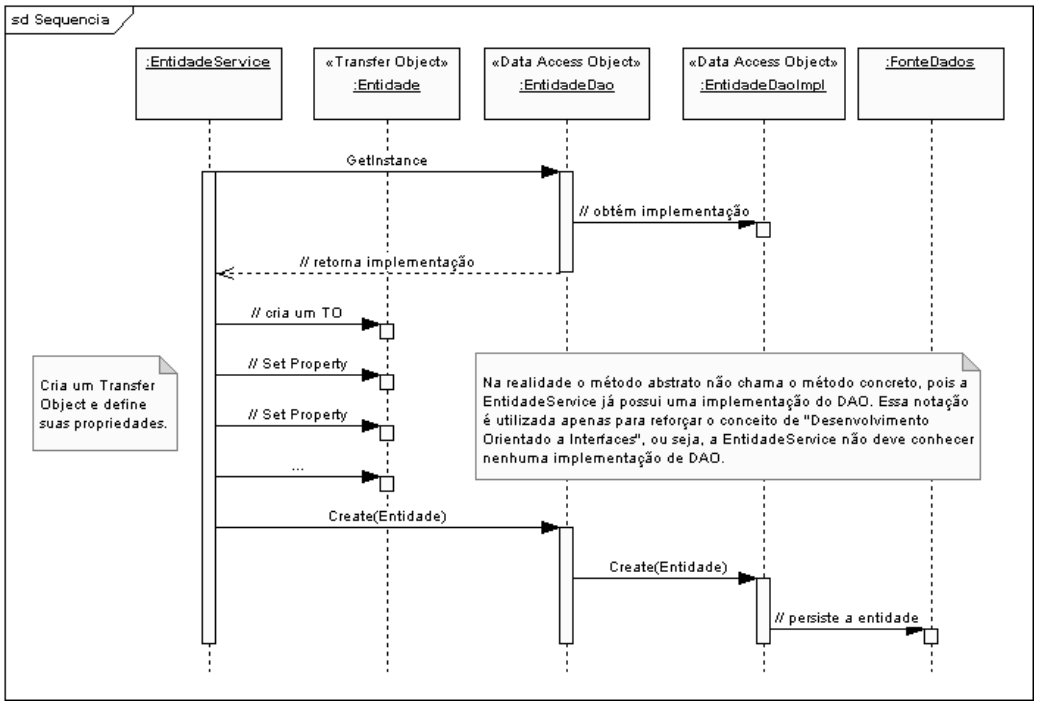

Figura 29: Interação entre elementos da camada de persistência

Por fim, a camada de integração é responsável pelos processos que demandam interação entre sistemas, através de mecanismo de mensagens usando o ESB (*Enterprise Service Bus*). Oferece um modelo desacoplado de integração e permite que diferentes sistemas conversem independente de protocolo ou formato de dados.

Os serviços desta camada também consomem os serviços disponibilizados na camada de interface de serviço. A camada de interface de serviços pode fazer uso dos serviços desta camada para automatizar um processo de negócio, por exemplo. A figura 26 ilustra estes serviços numa camada separada para enfatizar que os mesmos serão suportados pelo ESB, sendo mais uma opção de integração para a plataforma do nodo *gateway*. Estes serviços normalmente têm as seguintes características:

- Expõem funcionalidades dentro de um contexto de processamento específico;
- São desenhados com base em recursos disponíveis dentro de uma dada plataforma;
- São genéricos e reutilizáveis;
- Podem ser usados para integração ponto a ponto com outras aplicações;
- Podem ser formados pela composição de serviços desenvolvidos “em casa” e serviços de terceiros, comprados ou alugados.

Esta camada permite construir serviços compostos, serviços formados a partir de outros serviços, e também oferece os recursos necessários para adequar uma mensagem antes de invocar um serviço externo (ou interno), seja por diferença de protocolo, formato da mensagem ou formato de arquivo.

\

CAPÍTULO 6 – ESTUDO DE CASO

Atuação e Monitoramento Predial representam uma classe das aplicações de Redes de Sensores e Atuadores Sem Fio com enorme potencial de benefícios para a comunidade científica e a sociedade em geral. A fim de propor uma arquitetura e infraestrutura de hardware para os nodos sensores/atuadores e para o nodo sorvedouro, foram realizados estudos detalhados das principais arquiteturas de *hardware* – tipos de sensores, micro-controladores, módulos de comunicação sem fio, tipos de bateria para alimentação, servomecanismos, etc. – a ser adotada para o domínio de aplicação de monitoramento, atuação e segurança patrimonial. Para validar e consolidar os estudos efetuados, foi desenvolvido um Estudo de Caso cujo objetivo foi projetar um sistema baseado nas características de Redes de Sensores e Atuadores Sem Fio para prover atuação, monitoramento e segurança patrimonial. Por meio desse sistema um usuário poderá interagir remotamente, através de uma interface amigável, com dispositivos de *hardware* complexos, como nodos sensores e atuadores. Esta interação permite que o usuário do sistema capture dados do ambiente onde está instalada a Rede Sem Fio (temperatura, umidade do ambiente, etc), capture em tempo real *stream* de vídeo (através das câmeras de monitoração/vigilância), seja notificado (por e-mail, SMS, etc) da ocorrência de algum evento (detecção de intrusão, presença de determinado gás ou fumaça no período monitorado, etc) e atue modificando o estado de algum dispositivo (por exemplo, acendendo ou apagando uma lâmpada elétrica, ligando ou desligando um sistema de irrigação, controlando a temperatura do ambiente, etc). Como na maioria das vezes o usuário do sistema pode encontrar-se em trânsito e precisa acessar o sistema remotamente, devem ser disponibilizados interfaces de acesso, como por exemplo, através de um *smartphone* ou celular.

6.1 Estudos de tecnologias para os nodos sensores/atuadores e nodos sorvedouros

A arquitetura para a implementação de uma RSASF exige que diferentes plataformas de nodos sejam investigados, devido à discrepância de especializações e requisitos que a utilização de uma RSASF para determinado domínio de aplicação pode exigir. Por exemplo, para monitoração e controle de uma residência, seriam necessários vários tipos de plataformas, como nodos sensores com pouco poder de processamento e pouca largura de banda para comunicação para detecção de movimento, detecção da abertura de uma janela ou porta, etc. No entanto, em alguns casos, faz-se necessária a utilização de nodos sensores com maior poder de processamento e com maiores taxas de transmissão, como aqueles casos onde é necessário capturar vídeo (dados multimídia). Também podemos levantar diferenças em relação ao consumo de energia, seja pelo micro-controlador ou pela unidade de sensoriamento, que implicarão em plataformas diferentes. Já para os nodos sorvedouros, o alto poder de processamento (comparado ao que é exigido nos nodos sensores) e opções de diferentes formas de comunicação de alta largura de banda com o nodo *gateway* são pré-requisitos. As variáveis, nesse caso, passam a ser quantidade de memória disponível e custo financeiro, entre outras.

Devido aos recentes avanços na arquitetura de *hardware*, hoje se tem disponível uma quantidade enorme de opções com finalidades que irão ao encontro das necessidades específicas de cada nodo. Por exemplo, a utilização de uma RSASF para monitorar ou rastrear produtos ou ativos, pode utilizar dispositivos *Smart Dust* [67], que são sensores que possuem dimensões extremamente reduzidas, mas que podem detectar grandezas físicas tais como luz e vibrações. Devido aos recentes avanços na microeletrônica e nas técnicas de fabricação de *chips,* estes "motes" eventualmente podem possuir o tamanho de um grão de areia e ainda possuir diversas capacidades de sensoriamento, capacidade de computação e tecnologias de comunicação sem fio bidirecional. Esses *motes* podem capturar grandes quantidades de informações, processá-las e transmití-las a outros *motes* localizados a distancias de até 300 metros via comunicação sem fio [68]. A figura 30 ilustra um nodo sensor *SmartDust*, evidenciando seu reduzido tamanho físico.

Figura 30: Nodo sensor *Smart Dust*

As possibilidades da utilização desses *motes* em aplicações são inúmeras (indo além do rastreamento de objetos), podendo também rastrear pacientes dentro de um hospital ou detectar defeitos em equipamentos eletrônicos através da vibração excessiva de equipamentos industriais.

No entanto, a utilização dos *motes* para a utilização e comercialização em grande escala ainda está longe de acontecer, pois depende de pesquisas para redução do consumo de energia e melhorias nos protocolos de roteamento. Instituições como Universidade da Califórnia, Berkeley e Los Angeles, MIT, etc. trabalham em conjunto para ainda resolver esses problemas remanescentes. Atualmente, nodos sensores *Smart Dust* possuem também capacidade de comunicação ótica, tornando-os ainda menores (com dimensões aproximadas de 1,5 mm^3 e massa total de 5 mg).

Um exemplo de *hardware* da plataforma *Smart Dust* é o *Spec*, desenvolvida pela Universidade da Califórnia, Berkeley. Esse nodo foi projetado para ser de baixíssimo custo e operar com baixo consumo de energia, possuindo dimensões da ordem de 2.5 mm x 2.5 mm com unidade de processamento, memória RAM e capacidade de transmissão de dados (embora a curtas distâncias).

Voltado para aplicações onde a flexibilidade é pré-requisito ou é muito importante, o *Mica2* [69], também desenvolvido pela Universidade da Califórnia, Berkeley, inclui uma interface que permite conectar um conjunto de sensores e possui grande número de pinos de E/S e opções de expansão. Devido a essas características, o *Mica2* é uma opção adequada para qualquer aplicação onde custo e dimensões não sejam absolutamente críticos. A figura 31 ilustra um dispositivo *Mica2*.

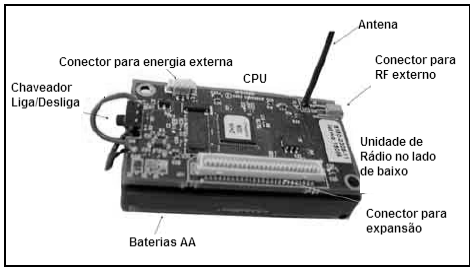

Figura 31: Plataforma *Mica2*, com o conector de expansão, permitindo conectar um vetor de sensores

O *Mica2* é capaz de receber mensagens de nodos sensores como o *Spec*, embutidos em ativos de relativo alto valor financeiro (produtos eletrônicos, por exemplo) passíveis de serem roubados. A memória e o poder de processamento do *Mica2* tornam esse nodo capaz de rastrear dezenas de nodos *Spec*. No entanto, o *Mica2* não é capaz de processar dados multimídia, como vídeo e som.

Para aplicações que precisam de mais recursos computacionais, tais como poder de processamento, memória RAM, comunicação, etc. a *Intel Research* disponibiliza algumas plataformas de hardware. A primeira delas é a *iMote* [70], capaz de processar dados multimídia, possui memória auxiliar Flash e transmissões que podem chegar a até 500Kbps. A última versão desta plataforma, a *iMote2* [71], possui processador baseado na arquitetura *PXA 271 XScale* com *clock* de 416MHz, memória de 256 kB SRAM, 32 MB de memória Flash e 32 MB de SRAM. Já a segunda plataforma de *hardware* da Intel para nodos de RSASF é a plataforma *Stargate* [72]que inclui um microprocessador também baseado na arquitetura X-scale com *clock* de 400MHz, 32 megabytes de memória RAM, interface de E/S universal para conexão com dispositivos através de interfaces serial, USB, Ethernet, Bluetooth, etc. A plataforma *Stargate* é capaz de conectar diretamente com dispositivos baseados na plataforma *Mica2* e *iMote* e interligar a rede formada por esses dispositivos a uma rede padrão como Ethernet, 802.11 e WANs (*Wide-Area Network*).

Para este trabalho, decidimos propor uma arquitetura de *hardware* diferente para os nodos sensores/atuadores e para os nodos sorvedouros. Essa arquitetura proposta foi norteada, principalmente, pelo fator financeiro. Para propor esta arquitetura, foram realizados inúmeros estudos de variados tipos de micro-controladores, microprocessadores, rádio modems, transceptores, etc. A arquitetura proposta será apresentada nas próximas seções deste capítulo.

6.1.1 Micro-controladores PIC

A Microhip é uma empresa precursora no uso de tecnologia RISC (*Reduced Instruction Set Computer*) em micro-controladores. Diferente da arquitetura de *von-Neumann* (CISC – *Complex Instruction Set Computer*), a estrutura RISC é baseada em barramentos independentes para dados e para programa com tamanho diferenciados. Por exemplo, no PIC16C55 o barramento de dados é de 8 bits, enquanto o de programa é de 14 bits, o que significa que uma instrução está "empacotada" em uma única palavra de programa (no caso do PIC16C55 de 14 bits) e, além de conter o *opcode* (instrução), contém também os operandos (dados para execução da instrução).

A Microchip oferece famílias de microcontroladores de 8 bits no barramento de dados que podem ser adaptados a uma gama de projetos. Dentre elas pode-se destacar:

- PIC12CXXX: linha compacta;

- PIC16C5X/PIC16C55X: linha-base;

- PIC16CXX: linha intermediária;

- PIC17CXX/PIC18CXX: topo de linha.

Todas as famílias oferecem diversas opções de memória de programa: OTP (*One Time Programmable*) e EPROM (*Erasable and Programmable Only Memory* – utilizada para desenvolvimento). Além disso, apresentam opções de baixa tensão e inúmeros tipos de circuito osciladores, assim como várias opções de encapsulamento. Alguns componentes estão disponíveis em memória somente de leitura (ROM mascaradas) e memória apenas de leitura programável e eletricamente apagável (EPROM/FLASH reprogramáveis).

Deve-se ressaltar que os micro-controladores da série PIC estão disponíveis em uma ampla gama de modelos para melhor se adaptarem às exigências de projetos específicos, diferenciando-se pelo numero de linhas de E/S e pelo conteúdo do dispositivo.

O alto desempenho da família de micro-controladores PIC pode ser atribuído às seguintes características de arquitetura RISC:

- mapa de registradores versátil;

- todas as instruções com palavras simples;

- palavra de instrução longa;

- arquitetura de instruções em "*pipeline*";

- instruções de apenas um ciclo de máquina;

- conjunto de instruções reduzido;

- conjunto de instruções ortogonal (simétrico).

O micro-controlador PIC 16F84a apresenta arquitetura do tipo *Harvard*, enquanto grande parte dos outros micro-controladores apresenta arquitetura *von-Neumann*. A diferença encontra-se na maneira como os dados são passados. Na arquitetura de *von-Neumann* é utilizado um único barramento (*bus*) interno, geralmente de 8 bits por onde passam as instruções e os dados. Já na arquitetura *Harvard,* existem dois barramentos: o barramento de dados (é sempre de 8 bits) e o barramento de instruções (é de 14 bits para o PIC 16F84a). Isso é útil, pois, enquanto uma instrução está sendo executada, outra já pode ser buscada na memória, tornando o processamento muito mais rápido. Além disso, como o barramento de instruções do PIC 16F84a é maior que 8 bits, a instrução já inclui o endereço do(s) operando(s) e o endereço de memória onde ela vai operar. Assim, apenas uma posição de memória é utilizada por instrução, otimizando o uso da memória de programa. Este diferencial no tamanho do barramento de instruções do PIC 16F84a deve-se à utilização da tecnologia RISC. Assim, o PIC 16F84a possui até 35 instruções (o número correto varia de acordo com o micro-controlador), menos que os micro-controladores convencionais que incorporam tecnologia CISC e que chegam a possuir mais de 100 instruções. Por ter essa característica, o PIC 16F84a facilita e dinamiza o aprendizado, porém é necessário saber que este conjunto mínimo de funções, em muitos casos, não atende completamente às necessidades do projeto. Assim, todas as funções não implementadas nos circuitos internos do micro-controlador devem ser implementadas pelo desenvolvedor.

As características básicas (gerais, de periféricos e elétricas) do micro-controlador PIC 16F84a são:

- Micro-controlador de 18 pinos, com encapsulamento plástico tipo DIP (*Dual In Line Package* ou Empacotamento em Duas Linhas);

- Quatro interrupções disponíveis (de TMR0, externa, por mudança de estado e de final de escrita na EEPROM);

- Memória de programação EEPROM FLASH, a qual permite a gravação do programa diversas vezes na mesma pastilha;

- Memória de dados RAM (*Random Access Memory* ou Memória de Acesso Aleatório) interna de 68 bytes, juntamente com uma memória de dados EEPROM (não-volátil) interna de 64 bytes;

- Barramento de instruções com 14 bits de largura e um conjunto de apenas 35 instruções (todas as instruções com um ciclo de máquina, exceto para desvios que levam dois ciclos de máquina);

- Barramento de dados de 8 bits;

- 16 registradores de funções especiais de hardware;

- Pilha (*stack*) de 13 bits com oito níveis de profundidade;

- Velocidade de operação de até 20 MHz de *clock*;

- Endereçamento nos modos direto, indireto e relativo para instruções e dados;

- Programa gravado em EEPROM, com até 1.000.000 de ciclos de apagamento e escrita, com retenção garantida por mais de 40 anos;

- 13 pinos de entrada e saída individualmente configurados;

- temporizador de 8 bits programável (TMR0) e contador com pré-divisor de 8 bits programável;

- *Power-On-Reset* (POR);

- Temporizador *WathDog Timer* (WDT) com oscilador próprio para operações seguras;

- Sistema de proteção de código programável na EEPROM;

- modo S*leep* para diminuição do consumo de energia;

- opções de oscilador selecionável (RS – oscilador RC de baixo custo, XT – cristal de quartzo padrão, HS – cristal de quartzo de alta velocidade e LP – cristal de quartzo de baixa freqüência (redução de consumo);

- programação serial "*in-circuit*" (por meio de dois pinos);

- 4 bytes de identificação (ID) programáveis pelo usuário;

- temperatura de trabalho entre -50ºC e + 125ºC;

- temperatura de armazenamento entre – 65ºC a + 150ºC;

- tensão de trabalho entre +2,0 e +6,0 Vcc;

- tensão máxima no pino Vdd (em relação ao Vss) entre -0,3Vcc e +7,5Vcc;

- tensão máxima no pino /MCLR (em relação ao Vss) entre -0,3 Vcc e + 14Vcc;

- tensão máxima nos demais pinos (em relação ao Vss) entre -0,6Vcc e (Vdd +0,6) Vcc;

- dissipação máxima de potência igual a 800 mW;

- corrente máxima de saída no pino Vss igual a 150 mA;

- corrente máxima de entrada no pino Vdd igual a 100 mA;

- corrente máxima de entrada de um pino (quando em Vss) igual a 24 mA;

- corrente máxima de saída de um pino (quando em Vdd) igual a 20 mA;

- corrente máxima de entrada do PORTA igual a 80 mA;

- corrente máxima de saída do PORTA igual a 50 mA;

- corrente máxima de entrada do PORTB igual a 150 mA;

- corrente máxima de saída do PORTB igual a 100 mA.

A figura 32 ilustra um micro-controlador PIC 16F84a, que será utilizado nos nodos sensores e atuadores neste trabalho.

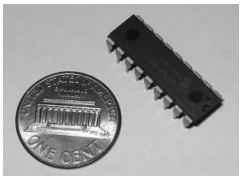

Figura 32: Micro-controlador PIC 16F84a

Uma das maiores vantagens do micro-controlador PIC16F84a é o seu baixo custo. Outro benefício, que chama bastante atenção dos desenvolvedores, é a simplicidade do conjunto de instruções. Além do mais, esse micro-controlador tem uma aplicabilidade muito ampla, podendo controlar desde sistemas de alarme residencial até sistemas de controle de produção em escala industrial.

Como desvantagens, este micro-controlador possui uma capacidade de memória muito limitada, cujo uso em grandes aplicações necessita de acesso a dispositivos de memória externa; não possui a comunicação serial implementada em hardware, o que obriga o desenvolvedor a projetar o controle de comunicação serial, assim como possui um número muito pequeno de portas E/S.

Entendem-se que os micro-controladores PIC 16F84a são adequados para constituírem a plataforma de *hardware* dos nodos sensores e atuadores. Os micro-controladores da série PIC foram escolhidos principalmente por ser um micro-controlador bastante utilizado e conhecido, de baixo custo e consumo de energia, possuir uma ótima IDE de desenvolvimento gratuita e por possuir muita documentação na Internet. O micro-controlador PIC 16F84a é um dispositivo computacional adequado para a experimentação e o desenvolvimento de pequenos e médios projetos, devido aos baixos custo e consumo, pequeno tamanho físico, facilidades de programação e manuseio, além de flexibilidade. Estas são algumas das exigências de fundamental importância na concepção e na implementação de projetos de sistemas micro-controlados. Logo, para as aplicações propostas neste trabalho, a escolha de um micro-controlador PIC levou em consideração a melhor relação custo/benefício, baixo custo de *hardware* e pequena ocupação de espaço físico. Com base nestas exigências, optou-se pelo micro-controlador PIC 16F84a pelo fato de ele pertencer a uma classe de micro-controladores de 8 bits, possuir uma arquitetura RISC genérica, ter uma boa relação custo/benefício e ter dimensões reduzidas.

6.1.2 Microprocessadores Rabbit

Os microprocessadores *Rabbit* da série 3000 e 3700 foram baseados em arquitetura do antigo Z180 e com um nível de integração baseados nos modernos microprocessadores, resultando no diagrama de bloco da figura 33.

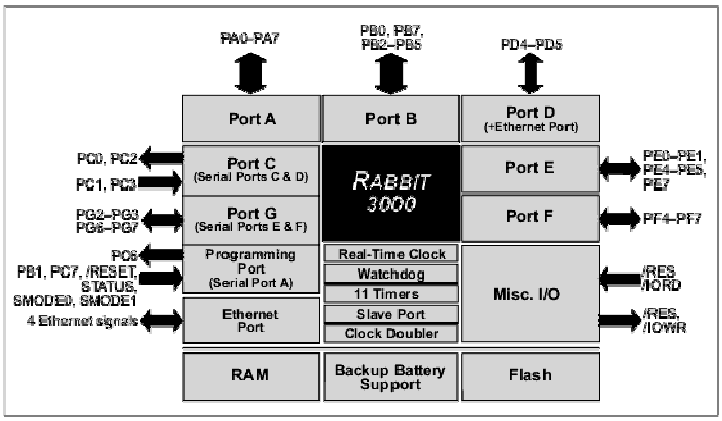

Figura 33: Diagrama de blocos do microprocessador *Rabbit 3000*

Os microprocessadores *Rabbit 3000* e *Rabbit 3700* são baseado em uma CPU com *clock* de 29.4 MHz e 22.1 MHz, respectivamente, portas de entrada e saída com várias funções secundárias, relógio de tempo real, *watchdog*, bateria de backup e eletrônica integrada para gerenciamento de memória flash, EPROM e RAM, ou seja, é um microprocessador que foi projeto pensando-se em automação.

Todo o projeto de hardware levou em consideração principalmente em se ter a linguagem de programação C ANSI, por isto, se utiliza de um XPC, *stackpointer* em *xmem* facilmente administrado por código C e instruções mistas 16/8 bits.

Com este planejamento obteve-se as seguintes vantagens:

- Arquitetura *glueless* facilitando circuitos de hardware;

- 6 portas seriais síncronas ou assíncronas;

- Precisão na geração de pulsos;

- Baixo nível de emissão de ruídos (*espectrum epeader);*

- Vários níveis de interrupção;

- Funcionamento em baixa freqüência (32KHz) ou alta velocidade (até 29.4 MHz);

- Processa até 50000 linhas de código C;

- Compatível com código Z80 / Z180 (90%);

- Gerencia códigos diretamente até 1 Mbit;

- Até 54 pinos de E/S, todos com funções secundárias;

- Até 11 *timers* programáveis;

- Relógio de tempo real incorporado no chip;

- *Watchdog* incorporado;

- Suporta IRDA, I2C, SPI;

- Excelente cálculo em ponto flutuante.

A figura 34 ilustra os dois modelos de microprocessadores *Rabbit* utilizados no Estudo de Caso.

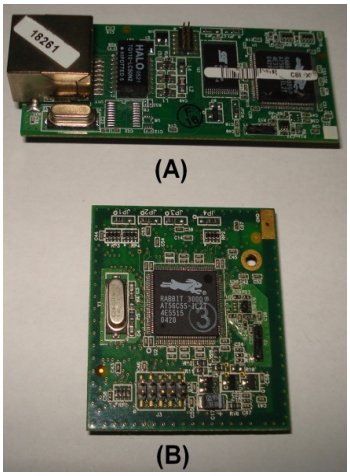

Figura 34: (A) Microprocessador *Rabbit 3700* com saída Ethernet; (B) Microprocessador *Rabbit* 3000 com entrada USB

O microprocessador *Rabbit 3700* é ideal para constituir o nodo sorvedouro. A sua interface padrão *Ethernet* possibilita que este nodo se conecte com dispositivos roteadores que utilizam tecnologia *xDSL, Cable Modem*, etc para interconectar à Internet.

Já o microprocessador *Rabbit 3000* é adequado para constituir os nodos sensores equipados com câmeras de vigilância/monitoramento, sendo que sua interface USB 2.0 permite conectar câmeras CMOS, amplamente encontradas no mercado, por baixo custo financeiro.

Dessa forma, neste trabalho propusemos a utilização dos micro-controladores PIC 16F84a para constituir os nodos sensores e atuadores, com exceção dos nodos de vigilância/monitoramento, que serão equipados com microprocessadores *Rabbit 3700* e câmeras CMOS. Já para os nodos sorvedouros propusemos a utilização dos microprocessadores *Rabbit 3000*.

6.1.3 Modems de Rádio Freqüência

A fim de se encontrar o modem de rádio freqüência com melhor custo beneficio para a comunicação entre o nodo sorvedouro e os nodos sensores/atuadores, foram estudados quatro tipos de dispositivos. A figura 35 ilustra os dispositivos que foram estudados e em seguida apresentamos as características de cada um. No final da seção, apresentaremos as justificativas para a escolha dos dispositivos que irão compor a arquitetura de hardware proposta neste trabalho.

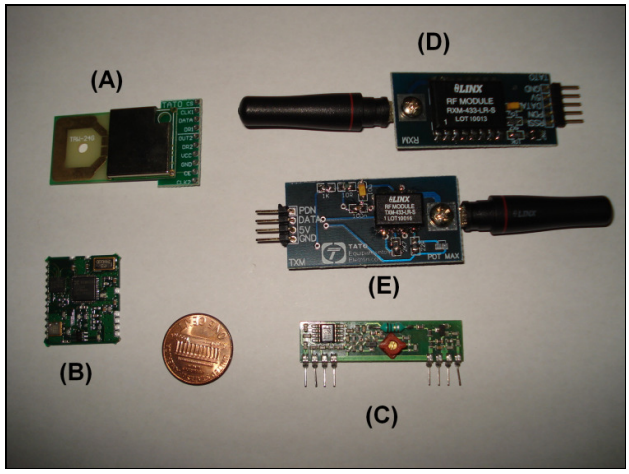

Figura 35: (A) Transceptor de 2.4 GHz; (B) Transceptor de 902-907.5 MHz; (C) Módulo Receptor/Transmissor de 433 MHz; (D) Módulo Receptor de 433 MHz; (E) Módulo Transmissor de 433 MHz

A seguir, faremos uma breve descrição dos dispositivos que irão prover a infraestrutura de enlace sem fio entre os nodos sensores/atuadores e o nodo sorvedouro que foram pesquisados e estudados neste trabalho.

(A) Transceptor Laipac TRW-2.4G: Freqüência de 2.4 GHz e Modulação GFSK. Taxas de transferência de 1 Mbps a 150m e 250 Kbps a 280m. Permite transmissão *full-duplex,* incluindo codificação, decodificação e *buffer* de dados.

(B) Transceptor Radiotronix Wi.232: O transceptor pode operar em dois modos, baixo consumo e DTS. No modo de baixo consumo, a saída do transmissor opera em 0 dBm e a máxima taxa de transmissão é de 38.4 Kbit/s. No modo DTS, a saída do transmissor é de + 12dBm e a taxa máxima de transmissão de dados é de 152.34 Kbit/s. Os dados transmitidos são validados através de CRC-16 internamente e codificados usando um algoritmo proprietário. Múltiplos módulos podem operar no mesmo canal devido ao seu protocolo CSMA/CA.

(C) Transceptor On Shine TRXA-433: Freqüência de 433 MHz e Modulação ASK. Taxa de transferência de 4800bps a 150m.

(D) e (E) Módulos Receptor e Transmissor Linx : Freqüência de 433 MHz, com taxas de transferências de 2400bps até 19200bps, com alcance de até 150m. Possui modo de economia de bateria (*Power Down*)

Os dispositivos que apresentaram melhor custo benefício no contexto da aplicação que foi desenvolvida neste trabalho foram o Transceptor Laipac TWR-2.4GHz - utilizados para transmissão dos dados multimídia capturados pelas câmeras CMOS, empacotados no microprocessador *Rabbit 3000* e transmitidos ao nodo sensor (*Rabbit 3700*) - e os módulos Receptor e Transmissor Linx, utilizados para comunicação através do enlace sem fio entre os nodos sensores/atuadores e o nodo sorvedouro.

A figura 36 ilustra a montagem de um nodo sensor capaz de detectar intrusão em determinada área e comunicar tal ocorrência ao nodo sorvedouro através de comunicação por rádio freqüência. Neste protótipo temos, dentre outros componentes, um nodo sensor, com

sua unidade controladora (um micro-controlador PIC16F84a), um dispositivo sensor de presença, bateria de alimentação e módulos TX e RX para estabelecer o canal de comunicação sem fio. Na mesma ilustração também apresentamos um protótipo de um nodo sorvedouro, com um microprocessador *Rabbit Série 3700,* capaz de coletar dados dos diversos nodos sensores, processá-los e enviá-los a um nodo gateway através de um *socket* TCP, utilizando tecnologia Ethernet e PPPoE.

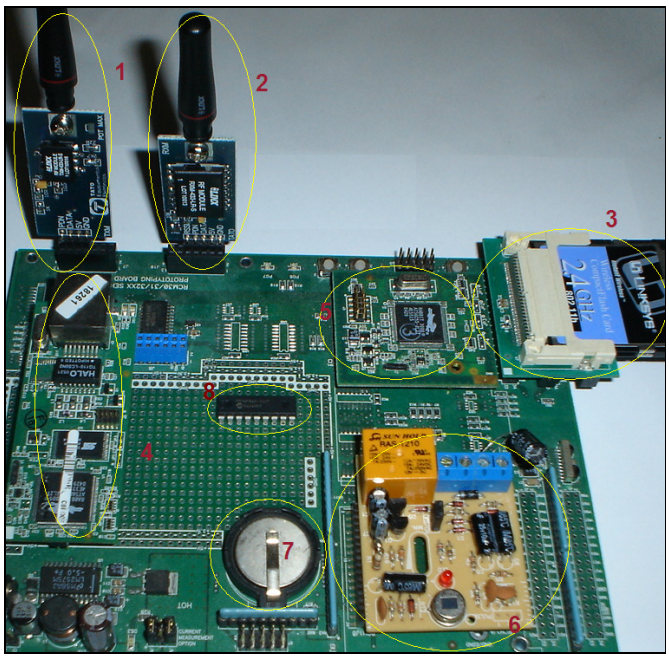

Figura 36: *Protoboard* para testes e construção de um nodo sensor e nodos sorvedouros

Os itens maçados na figuram referem-se a:

1) Módulo de comunicação por RF TX - (Transmissor);

2) Módulo de comunicação por RF RX – (Receptor);

3) Transceptor de comunicação por RF UHF, padrão 802.11b – WiFi. Este foi excluído dos testes devido a seu custo financeiro muito maior em relação aos demais dispositivos;

4) Microprocessador *Rabbit* Série 3700, representando o nodo sorvedouro;

5) Microprocessador *Rabbit* Série 3000 candidato a nodo sensor com câmera de vigilância/monitoramento;

6) Circuito integrado do sensor de presença;

7) Bateria para alimentação;

8) Micro-controlador PIC 16F84a candidato à unidade controladora do nodo sensor.

6.1.4 Custo Financeiro dos Componentes

A tabela 3 consolida o valor de cada componente de *hardware* estudado para o desenvolvimento dos nodos sensores e atuadores e do nodo sorvedouro.

Componente	Valor – US$
Micro-controlador PIC 16F84a	6,00
Microprocessador *Rabbit 3000*	25,00
Microprocessador *Rabbit 3700*	20,00
Transceptor Laipac TRW-2.4G	15,00
Transceptor Radiotronix Wi.232	60,00
Transceptor On Shine TRXA-433	5,00
Módulos Receptor e Transmissor Linx	5,00 (cada)

Tabela 4: Valor financeiro de cada componente estudado

6.2 Aplicações de Interface com o Usuário

Através das aplicações desenvolvidas nesse Estudo de Caso, um usuário poderá interagir remotamente, através de vários tipos de interfaces amigáveis, com dispositivos de *hardware* complexos, como nodos sensores e atuadores. Esta interação permite que o usuário do sistema capture dados do ambiente onde está instalada a Rede Sem Fio (temperatura, umidade do ambiente, etc), capture em tempo real *stream* de vídeo (através das câmeras de monitoração/vigilância equipadas com servo-motores que permitem movimentos *pan* e *tilt*), seja notificado (por e-mail, SMS, etc) da ocorrência de algum evento (detecção de intrusão, presença de determinado gás ou fumaça no período monitorado, etc) e atue modificando o estado de algum dispositivo (por exemplo, acendendo ou apagando uma lâmpada elétrica, ligando ou desligando um sistema de irrigação, controlando a temperatura do ambiente, etc). Como na maioria das vezes o usuário do sistema pode encontrar-se em trânsito e precisa acessar o sistema remotamente, devem ser disponibilizados diferentes tipos de interfaces de acesso, como por exemplo, através de um *smartphone* ou celular, além da interface Web.

6.2.1 Interface Web

A interface Web foi desenvolvida com tecnologia ASP.NET/C#. Para acessar o sistema o usuário deve fornecer seu *login* e senha e se autenticar no sistema. O serviço de segurança desenvolvido verifica a validade das informações e retorna uma lista das transações que o usuário tem acesso. O componente de Menu Dinâmico processa essa lista retornada e constrói o menu com as operações que o usuário tem direito de acesso.

Através da interface Web o usuário pode consultar informações coletadas pelos nodos sensores. Nesse estudo de caso, foram implementados sensores de temperatura do ambiente, de umidade do ar, detecção de intrusão e sensor de fumaça. Foram implementados também atuadores que podem interagir no ambiente, como por exemplo, ligando ou

desligando um dispositivo, tal como uma lâmpada. Por fim, foi desenvolvido um sistema para monitoração remota do ambiente, através de câmeras remoto-controladas através de servo-motores que permitem a rotação no eixo “X” e “Y” (movimentos pan e tilt).

Os vídeos capturados pelas câmeras de monitoração são decodificados na estação do usuário, através da tecnologia *Silverlight* [73]. A figura 37 ilustra a interface Web e a página onde o usuário pode controlar remotamente a câmera instalada na sua rede interna e monitorar o ambiente.

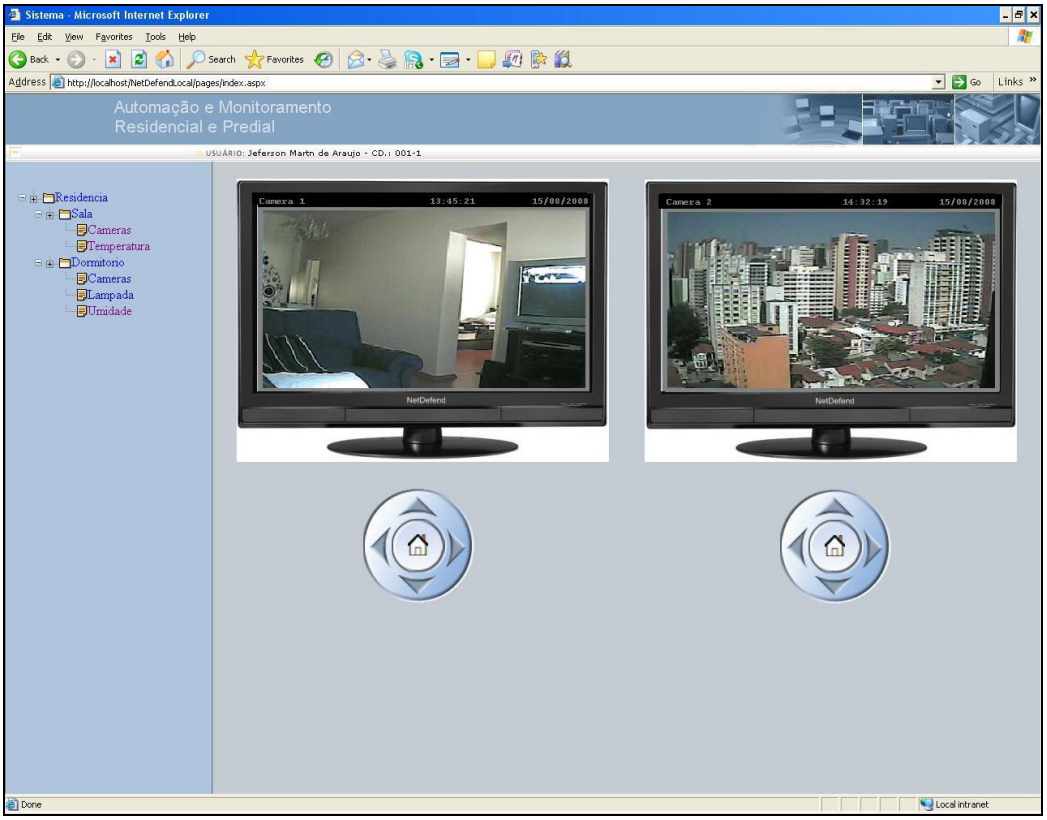

Figura 37: Interface Web. O vídeo capturado pelas câmeras de monitoração são decodificados na estação do cliente, através da tecnologia Silverlight

A tecnologia *Silverlight*é independente da plataforma do sistema operacional do cliente e compatível com os navegadores Web padrões do mercado.

6.2.2 Interface Móvel – Smartphone e celulares

Os usuários que se encontrarem em mobilidade e/ou não possuem acesso a um computador podem acessar o sistema e interagir com a sua rede interna através de *Smartphones* e telefones celulares. As aplicações instaladas nestes dispositivos comunicam-se com o nodo *gateway* através dos *Web Services* hospedados nesse nodo.

A aplicação instalada nos *Smartphones* habilitam os usuários a capturar informações dos nodos sensores, enviar comandos para os nodos atuadores e monitorar próximo de tempo real (o atraso do vídeo dependerá do tipo e da qualidade da conexão com a Internet) o ambiente onde está instalada a infraestrutura da sua rede interna, através das câmeras de monitoramento (as quais podem também ser remoto controladas pela aplicação). Já a aplicação instalada nos telefones celulares permite apenas que os usuários recebam informações coletadas pela sua RSASF e enviem comandos aos nodos atuadores. Devido às restrições do *hardware* e do *software* dos dispositivos hospedeiros é impraticável o envio e decodificação de *stream* de vídeo. No entanto, os usuários destes dispositivos podem capturar *snapshots* através das câmeras de monitoramento.

É através destes dispositivos que os usuário são notificados da ocorrência de determinado evento em uma região monitorada pelos nodos sensores, como por exemplo, a detecção de intrusão ou de fumaça. A notificação ocorre através do envio de mensagens SMS, disparadas pelo *Notification Service*, apresentado no quarto capítulo. A notificação também poderá ocorrer através de envio de e-mail, de acordo com a preferência do usuário.

As aplicações para os *Smartphones* foram desenvolvidas no *Microsoft .NET Compact Edition*3.5 e são compatíveis com os sistemas operacionais *Pocket PC* e *Windows Mobile*. Já as aplicações para telefones celulares foram desenvolvidas com o framework JME (*Java Micro Edition*) e podem ser instaladas e executadas a partir de qualquer dispositivo celular que possua a máquina virtual Java.

A figura 38 ilustra um simulador de *smartphone,* com sistema operacional Pocket PC 2003, com o aplicativo cliente atuando no ambiente da RSASF, através do envio de comandos para ligar/desligar uma lâmpada.

Figura 38: Aplicativo instalado em um simulador de *smartphone*, permitindo que o usuário interaja no ambiente de sua RSASF

6.2.3 Primitivas de Comunicação

Neste estudo de caso desenvolvemos alguns serviços de negócio para validar a viabilidade da arquitetura proposta e as tecnologias envolvidas. As primitivas abaixo foram utilizadas como interface para permitir a comunicação entre os domínios Usuário e o Nodo *Gateway*, entre o Nodo *Gateway* e o Nodo Sorvedouro e deste com os nodos sensores/atuadores. As tabelas 4, 5 e 6 consolidam as primitivas utilizadas para prover estas comunicações.

Comunicação	Método	Parâmetros
Usuário->Nodo *Gateway*	*getData*	*IDSinkNode, IDNode, GUID*
Usuário->Nodo *Gateway*	*setStatus*	*IDSinkNode, IDNode, StatusData, GUID*
Usuário->Nodo *Gateway*	*sendCommand*	*IDSinkNode, IDNode, CommandData, GUID*

Tabela 5: Primitivas para comunicação entre a aplicação cliente e o nodo *gateway*

Para se comunicar com o nodo *gateway,* a aplicação cliente sempre enviará o *GUID (Globally Unique Identifier)* que é um identificador retornado ao usuário sempre que este se autentica no sistema. Ao chamar um método remoto no nodo *gateway,* o sistema de segurança verifica se o GUID é válido (se não expirou, por exemplo), identifica o usuário e checa se ele tem autorização para executar o comando. Os campos *IDSinkNode* e *IDNode* são utilizados para identificar o nodo sorvedouro e os nodos sensores/atuadores dentro da RSASF. Não há necessidade destes campos representarem um identificador global único, já que essa identificação é válida apenas na RSASF que o usuário tem acesso. O comando *getData* é utilizado pela aplicação cliente para capturar informações dos nodos sensores. O comando *setStatus* é utilizado para gerenciar a unidade de sensoriamento dos nodos sensores, ligando ou desligando esta unidade. Por ultimo, o comando *sendCommand* é utilizado para enviar comandos aos nodos atuadores, como por exemplo, para rotacionar a câmera de vigilância no eixo "X" ou "Y".

Comunicação	Método	Parâmetros
Nodo *Gateway*->Nodo Sorvedouro	*getData*	*IDSinkNode, IDNode*
Nodo *Gateway*->Nodo Sorvedouro	*setStatus*	*IDSinkNode, IDNode, StatusData*
Nodo *Gateway*->Nodo Sorvedouro	*sendCommand*	*IDSinkNode, IDNode, CommandData*
Nodo Sorvedouro->Nodo *Gateway*	*sendNotification*	*IDSinkNode, IDNode, Information*

Tabela 6: Primitivas para comunicação entre a o Nodo *Gateway* e o Nodo Sorvedouro

O nodo *gateway* utiliza as primitivas *setStatus*e *sendCommand* para enviar comandos aos nodos sensores/atuadores e a primitiva *getData* para coletar informações de sensoriamento. Já o nodo sorvedouro envia notificações de eventos (capturados pelos nodos sensores) para o nodo *gateway* através da primitiva *sendNotification,* que contém a identificação do nodo sensor que capturou o evento (*IDNode*), a identificação do próprio nodo sorvedouro (*IDSinkNode*)e a informação sobre o evento detectado (*Information*).

Já a comunicação entre o nodo sorvedouro e os nodos sensores/atuadores é efetuada através das primitivas da tabela 6.

Comunicação	Comando	Parâmetros
Nodo *Sorvedouro*->Nodo Sensor/Atuador	*getData*	*IDSinkNode, IDNode*
Nodo *Sorvedouro*->Nodo Sensor/Atuador	*setStatus*	*IDSinkNode, IDNode, StatusData*
Nodo *Sorvedouro*->Nodo Sensor/Atuador	*sendCommand*	*IDSinkNode, IDNode, CommandData*
Nodo Sensor/Atuador ->Nodo Sorvedouro	*sendNotification*	*IDSinkNode, IDNode, Information*

Tabela 7: Primitivas para comunicação entre a o Nodo *Gateway* e o Nodo Sorvedouro

CAPÍTULO 7 – RESULTADOS, DISCUSSÕES E CONCLUSÃO

Diante da estruturação deste trabalho em uma jornada que combinou uma miríade de testes de validação, estudos dos conceitos de RSASF, estudos dos conceitos e tecnologias SOA e os conseqüentes resultados parciais obtidos, apresentados nos quatro primeiro capítulos, na proposição de uma arquitetura orientada a serviços para RSASF do capítulo 5 e no estudo de caso do capítulo 6, este capítulo final apresenta uma compilação dos resultados mais relevantes, o impacto em desempenho causado pela utilização de protocolos como HTTP e SOAP e da latência causada pela inclusão das camadas de virtualização de serviços, roteamento, ESB, etc inerentes dos mecanismos SOA.

7.1 Desempenho e escalabilidade de aplicações *Web Services*

De forma a permitir que aplicações clientes desenvolvidas com linguagens de programação diferentes e hospedadas em dispositivos com arquitetura de hardware e software diferentes possam acessar o mesmo ponto de publicação de um determinado serviço, optamos por desenvolver estes serviços através de uma tecnologia que seja capaz de balancear os benefícios de interoperabilidade entre plataformas diferentes sem agregar tamanha latência que impacte na interação homem-máquina.

Atualmente, a tecnologia de comunicação inter-processo que melhor atende essa necessidade é a implementação de *Web Services*. Nesse modelo de programação, geralmente é utilizado o protocolo SOAP para troca de informações e a linguagem WSDL é usada para descrever a interface desses serviços. Em tal infraestrutura, os *Web Services* podem ser acessados diretamente pelas aplicações clientes e podem ser combinados, formando uma aplicação composta.

A peça central na arquitetura *Web Services* é a linguagem XML, a qual permite a codificação de dados binários em um documento de texto. O protocolo SOAP é baseado na linguagem XML e é constituído de três elementos [74]: Envelope, *Header* e *Body*. O Envelope é o elemento raiz de uma mensagem, contendo um campo opcional, o *Header*, e um campo obrigatório, o *Body.* O elemento *Header* oferece uma estrutura flexível para especificar requisitos adicionais no nível da aplicação. Por exemplo, o elemento *Header* pode prover um mecanismo para gerenciamento de transações e autenticação/autorização. Já no elemento *Body* está contido o *payload*, ou seja, as informações efetivas de interesse da aplicação. Os serviços e operações providas são descritos na linguagem WSDL, que é baseada em *XML Schema*, provendo os elementos básicos para que uma aplicação cliente interaja com um *Web Service*, tais como localização do serviço e especificação do mecanismo de transporte utilizado para troca de mensagens. A arquitetura de *Web Services* é independente do protocolo de transporte, podendo utilizar protocolos tais como HTTP, SMTP, FTP, dentre outros.

Enquanto o protocolo SOAP não especifica qual o mecanismo de transporte deve ser utilizado na troca das mensagens entre o cliente e o provedor do serviço, a grande maioria das implementações SOAP que emulam o comportamento de RPC (*Remote Procedure Call*) através do par *request/reply*utilizam o protocolo HTTP.

As implementações do protocolo SOAP diferem entre si, principalmente no que diz respeito à facilidade de uso e ao desempenho. Atualmente, as duas principais tecnologias que implementam o padrão SOAP permitindo o desenvolvimento e hospedagem de aplicações orientadas a serviços através de *Web Services* são o .NET e o J2EE. Neste trabalho, foram desenvolvidas e disponibilizadas algumas interfaces para o usuário acessar o sistema, sendo que a maioria delas utiliza tecnologia *Web Services*. A tabela 7 consolida os tipos de interfaces disponibilizadas aos usuários do sistema, a tecnologia que dá suporte para a execução dessa interface no dispositivo hospedeiro, a tecnologia envolvida para essa aplicação cliente se comunicar com o nodo *gateway*e o protocolo de comunicação utilizado nessa comunicação é apresentada a seguir.

Interfaces Gráficas	*Plataforma*	*Tecnologia*	*Protocolo / Formatação*
Aplicações Desktop	.NET	Web Services	TCP / Formato Binário

Aplicações Web	Independente	Aplicação Web	HTTP
Aplicações Móveis Telefone Celular	J2ME	Web Services	HTTP / Formato SOAP
Aplicações Móveis *Smartphones* e PDA's	.NET	Web Services	HTTP / Formato SOAP

Tabela 8: Aplicações clientes, tecnologias e protocolos de comunicação envolvidos

Como podemos observar, a maioria das aplicações clientes se comunicam com os serviços através de *Web Services*. Dessa forma, é importante que o provedor do serviço seja desenvolvido utilizando uma tecnologia que melhor implemente o padrão SOAP em termos de escalabilidade e desempenho. Para isso, torna-se fundamental comparar o desempenho dos serviços implementados e hospedados com tecnologia .NET vs. tecnologia J2EE.

Assim, foram realizados dois testes. No primeiro teste, os *Web Services* e os serviços de negócio foram implementados e hospedados usando tecnologia .NET. No segundo teste, foi utilizada tecnologia J2EE para implementar os *Web Services* e os serviços de negócio e o IBM Web Sphere 6.1 para hospedá-los. Na tecnologia .NET, os *Web Services* comunicam-se com os serviços de negócio através do WCF. Por outro lado, na tecnologia J2EE, os *Web Services* comunicam com os serviços internos através de Java RMI [75]. Nos dois testes foram executados o métodos remoto *sendCommand*, introduzido no Capítulo 6.

O gráfico da figura 39 ilustra os resultados obtidos para a execução do experimento, que executou o método *sendCommand* iterativamente durante um período de 20 minutos, utilizando tecnologia .NET e tecnologia J2EE.

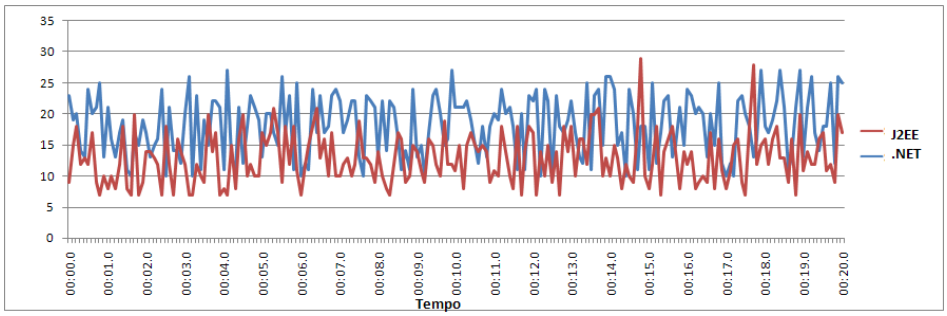

Figura 39: Métricas de desempenho (TPS) da aplicação desenvolvida e hospedada com tecnologia J2EE vs tecnologia .NET

Nestes experimentos, foi utilizada a forma de Conexão Direta (apresentada no quarto capítulo) sem utilizar a infraestrutura SOA, tais como os serviços de ESB (virtualização, roteamento, transformação de mensagem, verificação do contexto de segurança), de localização dos serviços através do *Service Registry*, etc. Pode-se notar que o desempenho da tecnologia .NET foi superior ao desempenho da tecnologia J2EE. A tecnologia .NET apresentou uma média de 18 TPS (Transações Por Segundo) enquanto que a aplicação desenvolvida e hospedada com tecnologia J2EE apresentou média de 13 TPS. Os resultados apresentados nessa seção justificam a escolha pela tecnologia .NET na proposição da arquitetura orientada a serviços.

As métricas de desempenho apresentadas na figura 39 representam os experimentos efetuados utilizando "Conexão Direta". No entanto, na proposição da arquitetura SOA, a forma comum de se acessar um serviço de negócio, tal como o serviço que prove o método *sendCommand,* é através da Conexão Intermediada (utilizando os mecanismos suplementares da arquitetura SOA). Dessa forma, faz-se necessário levantar as métricas de desempenho desse método através da Conexão Intermediada a fim de quantificar a sobrecarga adicionada em decorrência da inclusão dos mecanismos SOA. A figura 40 apresenta as evidências de desempenho para o mesmo experimento efetuado com a tecnologia .NET, alterando apenas a forma de conexão, que agora é Intermediada.

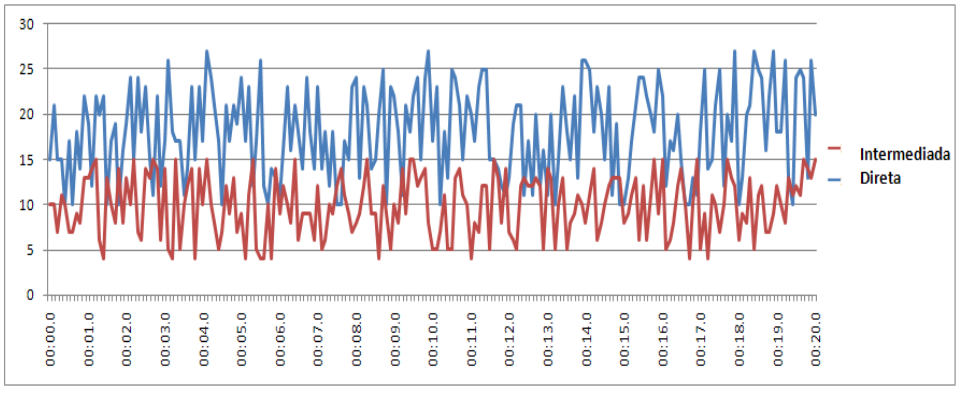

Figura 40: Métricas de desempenho (TPS) da aplicação desenvolvida e hospedada com tecnologia J2EE vs tecnologia .NET

Utilizando a conexão Intermediada a média foi de 10 TPS, uma sobrecarga de 55.55%, em média.

Para efetuar os experimentos foi utilizada a ferramenta *LoadRunner*[77], da Mercury/HP. Essa ferramenta foi configurada para enviar requisições ao nodo *gateway*até que o servidor atinja 100% de utilização dos seus processadores. Os resultados gerados foram consolidados no Excel para gerar os gráficos com as métricas de desempenho.

7.2 Desempenho e escalabilidade dos serviços *Multimídia Data Broker / Multimídia Data Entry e Multimídia Data Service*

Dentre os serviços de negócio, o que mais consome recurso computacional de CPU é o serviço *Multimídia Data Service.* Como apresentado no Capítulo 4, este serviço é responsável por codificar o *stream*de dados (enviado pelo Nodo Sorvedouro ao *Multimídia Data Entry*e destepara o *Multimidia Data Service*) em um formato padrão que possa ser decodificado pelos clientes. De forma a garantir a qualidade do vídeo, deve-se balancear a taxa de transferência de bits, ou *bit rate*, com a largura de banda disponível entre o nodo sorvedouro e o nodo *gateway* e deste com os clientes, supondo que a conexão entre cada um destes domínios ocorre através de uma rede pública como a Internet. Outro ponto importante a ser analisado em relação ao *Multimídia Data Service* é o planejamento de capacidade, ou seja, quantos *stream* de vídeos (advindos de diferentes fontes) esse serviço é capaz de atender (codificar) hospedado em um servidor com uma determinada configuração de hardware.

Essas duas análises serão exploradas em seguida.

7.2.1 Análise da taxa de transferência vs latência

Para sistemas de vigilância e segurança patrimonial é comum ajustar a codificação do vídeo para uma taxa de transferência de 259 Kbps ou 102 Kbps, com taxa de

quadros por segundo (*frame rate*, ou *frames per second*) igual a 0.5 fps. Para a primeira taxa, conseguem-se quadros com tamanho de 320 x 240. Já para a segunda taxa, o tamanho dos quadros diminui para 192 x 144.

Foram realizados dois experimentos – um experimento utilizando a taxa de transferência de 259 Kbps e o outro, 102 Kbps - a fim de dimensionar a latência para recebimento dos quadros. Nos dois experimentos, o link de comunicação do nodo sorvedouro com o nodo *gateway* e deste para a aplicação que decodifica o vídeo foi de 512 Kbps. O procedimento do teste foi bem simples. Em uma RSASF foram hospedados dois nodos: um nodo sensor para capturar vídeo do ambiente e um nodo atuador responsável por ligar e desligar uma lâmpada. A partir de uma interface Web, pôde-se observar o vídeo capturado e interagir com o ambiente ligando e desligando a lâmpada.

Primeiramente, foi coletado qual o tempo que o comando emitido pela aplicação Web consumiu para ser efetivamente executado, ou seja, quanto tempo o simples comando de ligar/desligar a lâmpada necessita para chegar ao nodo atuador e este efetuar a operação. Foram executadas 20 iterações deste mesmo comando e a média obtida foi de 375 ms.

Assim, no primeiro experimento, utilizando uma taxa de transferência de 102 Kbps para o vídeo, foi enviado o comando de ligar a lâmpada. O evento foi observado no vídeo após 4,543 s, em média (foram executadas 20 iterações). Subtraindo o tempo necessário para o comando chegar ao nodo atuador e este efetivamente acender a lâmpada, temos uma média de 4,168 s. Este é o tempo médio de latência, ou o tempo de atraso.

No segundo experimento, variou-se apenas a taxa de transferência, configurada para 259 Kbps. Com esta taxa, o tempo médio de latência observado foi de 17,322 s.

Portanto, pode-se concluir que a taxa de transferência é um fator de extrema importância e deve ser configurado criteriosamente, baseado nos pré-requisitos exigidos pelo domínio da aplicação.

7.2.2 Análise dos serviços de recebimento/publicação e de codificação/decodificação de vídeo

Os serviços *Multimidia Data Entry* e *Data Broker* (responsáveis por receber e difundir o *stream* de vídeo, respectivamente) juntamente com o serviço *Multimidia Data Service* são os serviços desenvolvidos para o nodo g*ateway* que mais demandam recurso computacional de CPU. Assim, faz-se necessário o estudo e planejamento de capacidade destes serviços de modo que a demanda não ultrapasse os recursos disponíveis, interferindo no comportamento do sistema como um todo e degradando a qualidade do serviço em si.

Para o serviço *Multimidia Data Service,* a fim de identificar o consumo de recursos demandado por esse serviço em relação ao número de fontes difundido *stream* de vídeo foram realizados alguns experimentos. O objetivo principal destes experimentos foi de verificar quantas fontes de vídeo simultâneas os serviços podem processar, baseado em uma configuração de hardware.

Desse modo, o serviço *Multimidia Data Service* foi hospedado exclusivamente em um servidor, com 4GB de memória RAM e 1 processador Intel Pentium IV Core2 Quad (4 núcleos). Os experimentos foram feitos, consecutivamente, com uma, duas, três, quatro e 18 fontes de vídeo. As figuras 41, 42, 43, 44 e 45 apresentam os dados capturados e consolidados através do utilitário *Performance Monitor* do Windows. O tempo de duração de cada experimento foi de 20 minutos.

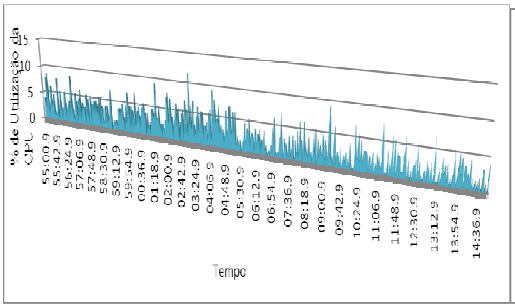

Figura 41: Consumo de CPU x Tempo para 1 fonte

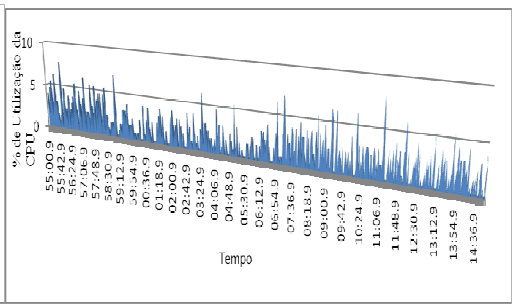

Figura 42: Consumo de CPU x Tempo para 2 fontes

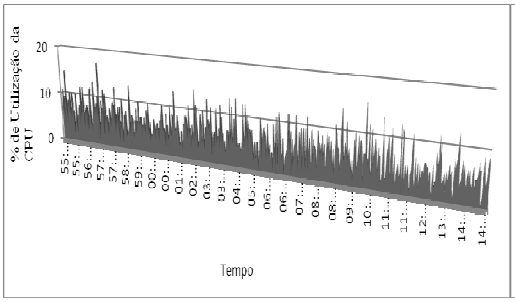

Figura 43: Consumo de CPU x Tempo para 3 fontes

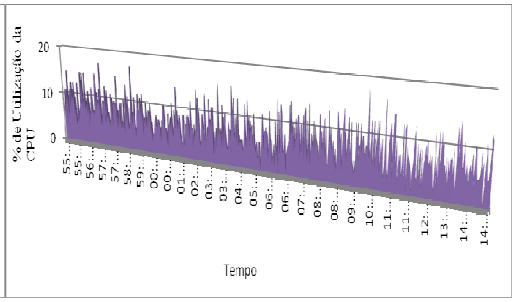

Figura 44: Consumo de CPU x Tempo para 4 fontes

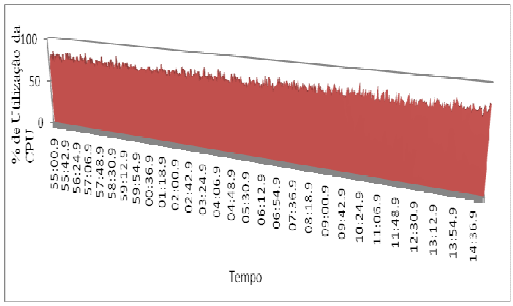

Figura 45: Consumo de CPU x Tempo para 18 fontes

A partir da análise dos dados de consumo de CPU capturados, pode-se concluir que o serviço *Multimídia Data Service* pode processar com segurança um máximo de 18 *streams* de vídeo simultaneamente. Para aumentar o número de fontes de dados multimídia, devem-se adicionar novos servidores (escalamento horizontal) de forma a balancear as requisições ou aumentar os recursos computacionais, adicionando processadores (escalamento vertical) à configuração de *hardware* na qual os experimentos foram realizados.

Casos em que são utilizados mais do que 10 fontes de dados, recomenda-se que o serviço *Multimida Data Service* seja hospedado em um servidor exclusivo, para não interferir nos demais serviços e prejudicar o sistema como um todo.

Já os serviços *Multimidia Data Entry* e *Multimidia Data Broker* foram hospedados em um servidor com 4GB de memória RAM e 1 processador Intel Pentium IV Core2 Duo (2 núcleos) para a realização dos experimentos que visão obter o consumo de CPU para difundir um número variável de fontes de dados simultaneamente a um único usuário. Os experimentos, executados no período de 20 minutos, foram realizados com uma, duas, três, quatro e 18 fontes de vídeo. As figuras 46, 47, 48, 49 e 50 apresentam os dados capturados e consolidados através do utilitário *Performance Monitor* do Windows.

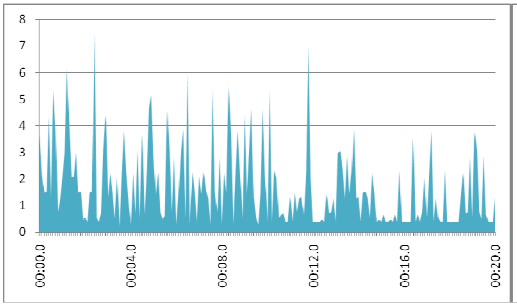

Figura 46:Consumo de CPU x Tempo para 1 fonte

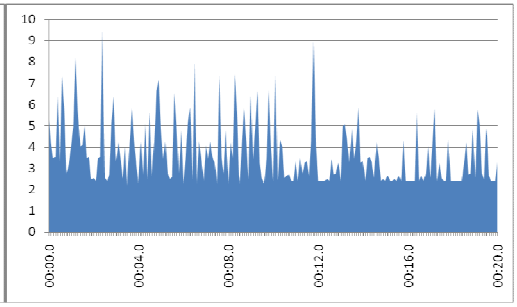

Figura 47: Consumo de CPU x Tempo para 2 fontes

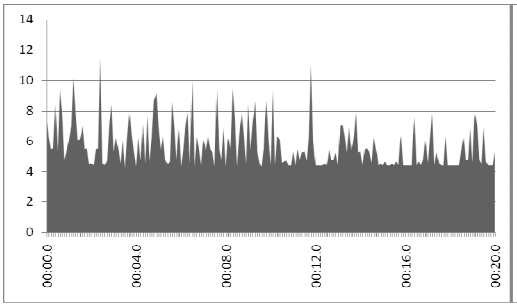

Figura 48: Consumo de CPU x Tempo para 3 fontes

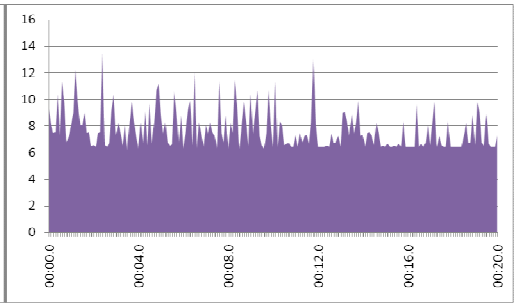

Figura 49: Consumo de CPU x Tempo para 4 fontes

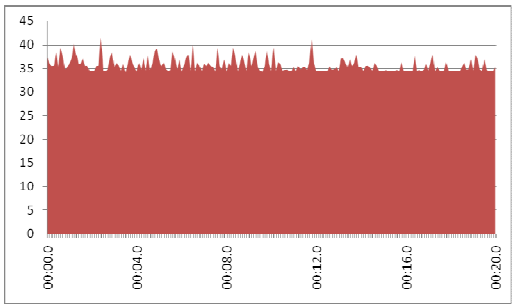

Figura 50: Consumo de CPU x Tempo para 18 fontes

Como podemos observar, os serviços de *Multimidia Data Entry* e *Data Broker* requerem menos processamento do que o serviço *Multimidia Data Service*. Assim, para até 18

fontes de fluxo multimídia, podemos considerar a abordagem de instalar este serviço de negócio juntamente com os serviços de infraestrutura genéricos da RSASF.

7.3 Análise do nodo sorvedouro para difusão do *stream* de vídeo

O nodo sorvedouro possui três interfaces para comunicação com outros dispositivos. Uma interface é utilizada para comunicação com o nodo *gateway*através da pilha de protocolos TCP/IP. As outras duas interfaces, as quais provêm dois links de comunicação sem fio, são utilizadas para comunicação com os nodos sensores e atuadores. Decidiu-se estabelecer dois links de comunicação entre os nodos sensores/atuadores e o nodo sorvedouro, sendo um link exclusivo – tipicamente com baixa largura de banda (2.4 Kbps) - para enviar informações relativas a comandos aos nodos atuadores e receber informações dos nodos sensores e o outro link exclusivo para recebimento do fluxo multimídia, capturado pelas câmeras de vigilância/monitoramento, empacotado e transmitido ao nodo sorvedouro pelos nodos sensores. Esta última interface geralmente possui largura de banda de até 1 Mbps. Dessa forma, é de extrema importância analisar a quantidade máxima de fontes de vídeo que a interface suporta sem gerar muito atraso na transmissão do fluxo.

Outro fator importante e que deve ser analisado é a capacidade de processamento do nodo sorvedouro. O microprocessador utilizado neste nodo possui uma CPU com *clock* de 22.1 MHz, que é um fator limitante para determinar o número máximo de pacotes que o nodo sorvedouro pode receber e transmitir ao nodo *gateway*.

Dessa forma, é necessário realizar experimentos a fim de levantar a capacidade de processamento do nodo sorvedouro e a capacidade do enlace sem fio utilizado para transmissão dos dados multimídia. As próximas seções apresentam os resultados obtidos a partir dos experimentos.

7.3.1 Capacidade de processamento do Nodo Sorvedouro

O nodo sorvedouro possui um microprocessador da família *Rabbit 3700* com *clock* de processamento de 22.1 MHz, apresentando, assim, recursos computacionais bastante limitados e que impactam diretamente na quantidade de nodos sensores que podem conectar-se ao mesmo nodo sorvedouro. Como a operação de transmissão de dados multimídia é a que mais consome processamento do nodo sorvedouro, realizamos alguns experimentos com o objetivo de planejar a capacidade deste nodo. Para verificar o comportamento da CPU do microprocessador na medida em que a carga aumenta (ou seja, novas fontes de vídeo são adicionadas) foi utilizada a aplicação *RabbitWeb* que disponibiliza *dashboards* com informações úteis sobre o consumo de CPU do microprocessador. O gráfico da figura 51 ilustra o consumo de CPU no decorrer de 3 horas. A cada 30 minutos foi incluída uma nova fonte de dados multimídia, que iniciou o trabalho de captura de vídeo e de transmissão do fluxo coletado para o Nodo Sorvedouro.

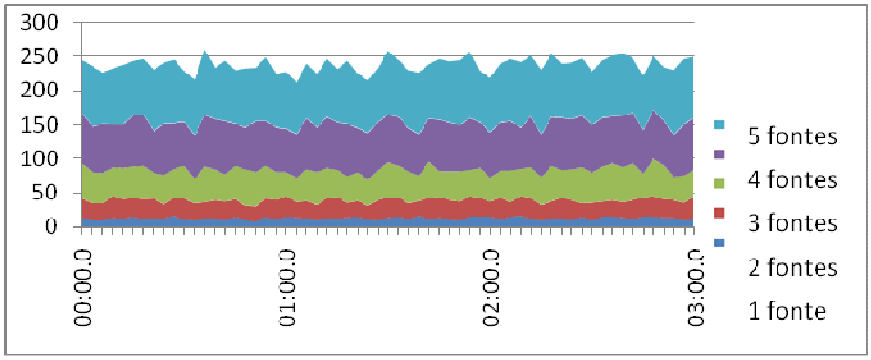

Figura 51: Consumo de CPU do nodo sorvedouro x Tempo (horas)

7.3.2 Capacidade do enlace de dados multimídia

O enlace de dados entre o nodo sorvedouro e os nodos com câmera de vigilância/monitoramento que foi proposto para utilização no sistema possui capacidade máxima de 1 Mbps. Portanto, além da capacidade de processamento da CPU do nodo

sorvedouro, esse canal de transmissão é outro fator limitante da quantidade de câmeras que podem se comunicar com um único nodo sorvedouro. Foi realizado um experimento para medir o consumo de banda consumido por uma fonte e para determinar a capacidade do link em decorrência do número de fontes de vídeo. Nesse experimento, foi verificado que a largura de banda média necessária para transmitir um *stream* de vídeo, configurado com uma taxa de transferência de 102 Kbps, pelo enlace sem fio entre o nodo sensor e o nodo sorvedouro é de 160 Kbps, utilizando o *transceiver* apresentado no sexto capítulo, exclusivo para o envio de dados multimídia.

7.4 Trabalhos Futuros

Uma possibilidade de projeto futuro está relacionado ao desenvolvimento de *Web Services* no padrão DPWS (*Devices Profile for Web Services*) [78] para o hardware dos microprocessadores *Rabbit* utilizados no nodo sorvedouro. O *framework* DPWS permite incluir funcionalidades SOA (tais como descoberta de serviços) e estende a capacidade dos *Web Services* através do padrão WS-* em dispositivos com recursos computacionais limitados.

7.5 Conclusão

A tecnologia das Redes de Sensores e Atuadores Sem Fio vem evoluindo exponencialmente nos últimos anos, fazendo com que o número de pesquisas científicas cresça em ritmo acelerado. O interesse nessa área deve-se ao fato dos inúmeros desafios ainda incipientes e da enorme oportunidade de aplicações práticas em campos e áreas diversos.

Por outro lado, é fato que a interação humana com a tecnologia também cresceu nas últimas décadas e é evidente a importância da tecnologia da informação na vida cotidiana das pessoas. Com o desenvolvimento de novas tecnologias e a evolução dos sistemas atuais, aplicações que antes eram tidas como futurísticas passarão a estar disponíveis e ao alcance dos cidadãos. Uma dessas aplicações é a utilização de uma RSASF para atuação, monitoramento e segurança patrimonial.

Nesse trabalho, propusemo-nos a estudar os componentes de uma RSASF, comparando as plataformas, arquiteturas e infraestruturas existentes a fim de propor uma arquitetura orientada a serviços para hospedar uma Rede Sem Fio que possa prover serviços no contexto de atuação e monitoramento com fins de vigilância patrimonial e que, ao mesmo tempo, seja flexível suficiente para atender outros domínios de aplicação, como monitoramento ambiental, segurança pública, rastreamento e controle autônomo de animais, etc. Como foi visto, o desenvolvimento de aplicações para RSASF pode obter vantagens quando estas aplicações são hospedadas em uma arquitetura SOA, com suporte a reusabilidade, manutenibilidade e estensibilidade.

Para alcançar este resultado, a comparação com o que já existe de mais moderno atualmente é fundamental. Para isso, fizemos um levantamento bibliográfico minucioso sobre o estado da arte no campo das RSASF e suas aplicações e propusemos uma metodologia criteriosa para avançar no sentido de obter um sistema que seja mais viável economicamente, mas ao mesmo tempo atendendo aos pré-requisitos que uma aplicação de atuação e monitoramento para segurança patrimonial exige, tais como desempenho, estabilidade, alta disponibilidade, etc.

As principais contribuições do sistema proposto podem ser sintetizadas sob dois aspectos. Do ponto de vista de projeto, a proposta oferece um paradigma orientado a serviços para a arquitetura de Redes Sem Fio de Monitoração e Atuação, baseada na área de *Web Services*, fornecendo todos os benefícios de flexibilidade e interoperabilidade inerentes a essa abordagem. Do ponto de vista de um sistema de *middleware*, a proposta procura incorporar todas as características e funcionalidades necessárias, tanto para as aplicações clientes como para os desenvolvedores dessas aplicações, livrando-os da tarefa de lidar com decisões de infraestrutura e assim facilitando o desenvolvimento de novos serviços e novas aplicações em outros domínios de aplicação. Este trabalho também contribuiu para a

comunidade científica propondo um novo *framework* com componentes genéricos de infraestrutura para o nodo *gateway*.

REFERÊNCIAS

[1] IEEE Std 1471-2000 IEEE Recommended Practice for Architectural Description of Software-Intensive Systems –Description. http://standards.ieee.org/reading/ieee/std_public/description/se/1471-2000_desc.html - Verificado em 14/05/2008.

[2] L. Bass,P. Clements, R. Kazman. *Software Architecture in practice*.Addison Wesley, 2 Edição, 2005.

[3] C. MacKenzie, K. Laskey, F. McCabe, P. Brown, R. Metz. *Oasis Reference Model for Service Oriented Architecture 1.0*.2006.

[4] M. Endrei, J. Ang, A. Arsanjani, S. Chua, P. Comte, P. Krogdahl, M Luo, T. Newling. *Patterns - Service-Oriented Architecture and Web Services*. IBM Redbook, 2004.

[5] A. Perrig, J. Stankovic, D. Wagner. *Security in Wireless Sensor Networks*.Communications of the ACM. p. 53-57, Volume 47, Issue 6, June 2004, New York, USA.

[6] Center for Future Health – Smart Medical Home - University of Rochester, New York.http://www.futurehealth.rochester.edu/smart%5Fhome– Verificado em 23/05/2007

[7] L. Schwiebert, S. Gupta, J. Weinmann. *Research Challenges in Wireless Networks of Biomedical Sensors*. Proceedings of the 7th annual international conference on Mobile computing and networking.p. 151-165, 2001, Rome, Italy.

[8] A. Mainwarning, D. Culler, J. Polastre, R. Szewczyk, J. Anderson. *Wireless Sensor Networks for Habitat Monitoring*.Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications.p. 88-97, 2002, Atlanta, Georgia, USA.

[9] T. Wark, C Crossman, W. Hu, Y. Guo, P. Valencia, P Sikka, P. Corke, C. Lee, J. Henshall, K. Prayaga, J. O´Grady, M. Reed, A. Fisher. *The Design and Evaluation of a Mobile Sensor/Actuator Network for Autonomous Animal Control*. Proceedings of the 6th International Conference on Information Processing in Sensor Networks. p. 206-215, 2007, Cambridge, Massachusetts, USA.

[10]M. Li, Y. Liu.*Underground Structure Monitoring with Wireless Sensor Networks*.Proceedings of the 6th international conference on Information processing in sensor networks.p. 69-78, 2007, Cambridge, Massachusetts, USA.

[11] M. Heikkilä, M. Pietikäinen. *An image mosaicing module for wide-area surveillance*.Proceedings of the third ACM international workshop on Video surveillance & sensor networks.p. 11-18, 2005, Hilton, Singapore.

[12] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, J. Zhao. *Habitat monitoring: Application driver for wireless communications technology*. Proceedings of the First ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean.ACM Press, 2001, New York, NY, USA.

[13] A. Loureiro, L. Ruiz, J. Nogueira. *Rede de sensores sem Fio*. Simpósio Brasileiro deComputação, Jornada de Atualização de Informática. p. 193-234. 2002.

[14] M. Ilyas, I. Mahgoub. *Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems*. CRCPress, 2005.

[15] National Science Foundation.*Report of the National Science Foundation Workshop on Fundamental Research in Networking*.Abril 24–25, 2003. Disponível emhttp://www.cs.virginia.edu/~jorg/workshop1 - Verificado em 09/05/2008

[16] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwarning, D. Estrin. *Habitat Monitoring with Sensor Networks*. Communications of the ACM. p. 34-40, Volume 47, Issue 6, June 2004, New York, USA.

[17] A. Loureiro, L. Ruiz, J. Nogueira. *Rede de sensores sem Fio*. Simpósio Brasileiro deComputação, Jornada de Atualização de Informática. p. 193-234. 2002.

[18] Microchip Technology, *PIC 16 MCU. http://www.microchip.com/ParamChartSearch/chart.aspx?branchID=1002&mid=10&lang=en&pageId=74* – Verificado em 06/07/2008.

[19] Radiotronix, http://www.radiotronix.com/products/proddb.asp?ProdID=187 – Verficado em 06/07/2008.

[20] Tato, http://www.tato.ind.br/ - Verificado em 06/07/2008.

[21] Rabbit Semiconductor, *Rabbit 3000 Microprocessors. http://www.rabbitsemiconductor.com/products/rab3000/index.shtml* - Verificado em 06/07/2008

[22] M. Ilyas, I. Mahgoub. *Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems*. CRCPress, 2005.

[23] K. Hwang, J. In, N Park, D. Eom.*A design and implementation of wireless sensor gateway for efficient querying and managing through World Wide Web.*IEEE Transactions on Consumer Eletronics. p. 1090-1097, Volume 49, Issue 4, Nov 2003, South Korea.

[24] M. Li, Y. Liu.*Underground Structure Monitoring with Wireless Sensor Networks*.Proceedings of the 6th international conference on Information processing in sensor networks.p. 69-78, 2007, Cambridge, Massachusetts, USA.

[25] Crossbow Technology.*Wireless Sensor Networks: Mica2*. http://www.xbow.com/Products/productsdetails.aspx?sid=174 – Verificado em 21/06/2008

[26] A. Mainwarning, D. Culler, J. Polastre, R. Szewczyk, J. Anderson. *Wireless Sensor Networks for Habitat Monitoring*.Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications.p. 88-97, 2002, Atlanta, Georgia, USA.

[27] T. Wark, C Crossman, W. Hu, Y. Guo, P. Valencia, P Sikka, P. Corke, C. Lee, J. Henshall, K. Prayaga, J. O´Grady, M. Reed, A. Fisher. *The Design and Evaluation of a Mobile Sensor/Actuator Network for Autonomous Animal Control*. Proceedings of the 6th International Conference on Information Processing in Sensor Networks. p. 206-215, 2007, Cambridge, Massachusetts, USA.

[28] L. Schwiebert, S. Gupta, J. Weinmann. *Research Challenges in Wireless Networks of Biomedical Sensors*. Proceedings of the 7th annual international conference on Mobile computing and networking.p. 151-165, 2001, Rome, Italy.

[29] *Eletronic Privacy Information Center (EPIC) Video Surveillance Information Page.*http://www.epic.org/privacy/surveillance/ - Verificado em 21/06/2008

[30] S. Hsu, D.C. *Forms Network of Surveillance*. Washington Post, Feb. 17, 2002.

[31] WebSite da Prefeitura da cidade de São Paulo. http://www.prefeitura.sp.gov.br/portal/a_cidade/noticias/index.php?p=14756 – Verificado em 21/06/2008

[32] *NYC Surveillance Camera Project*. http://www.mediaeater.com/cameras/ - Verificado em 23/06/2008

[33] Casa inteligente do Massachusetts Institute of Technologyhttp://architecture.mit.edu/house_n/index.html - Verificado em 25/06/2008

[34] *Desmistificando a Robótica.* http://bolzani.com.br/- Verificado em 01/07/2008

[35] R. Silva. *Suporte ao desenvolvimento e uso de frameworks e componentes*. Dissertação de Doutorado em Ciência da Computação, Universidade Federal do Rio Grande do Sul, Porto Alegre, Brasil, 2000.

[36] C. Larman. *Utilizando UML e Padrões*. Bookman, Porto Alegre, 2000.

[37] Microsoft Corporation, *.NET Framework Home*, http://msdn2.microsoft.com/pt-br/netframework/default.aspx - Verificado em 31/07/2008

[38] T. Ta, N. Othman, R. Glitho, F. Khendek. *Using Web Services for Bridging End-User Applications and Wireless Sensor Networks*. Proceedings of the 11th IEEE Symposium on Computers and Communications.p. 347-352, 2006, Washington, DC, USA.

[39] Sun Microsystems, *Java ME at a Glance,*http://java.sun.com/javame/index.jsp - Verificado em 31/07/2007

[40] F. Buschmann. *Pattern Oriented Software Architecture: A System of Patterns*. Wiley, 1996.

[41] L. Bass,P. Clements, R. Kazman. *Software Architecture in practice.*Addison Wesley, 2 Edição, 2005.

[42] H. Zimmermann. *OSI Reference Model – The ISO Model of Architecture for Open Systems Interconnection*.IEEE Transactions on Communications, vol. 28, no 4, p. 425-432, April 1980, USA.

[43] J. Bueno; P. Corrêa; Y. Onoe. et al. *Modelo de Referência para Arquitetura Orientada a Serviço.*OASIS, 2007. http://www.pcs.usp.br/~pcs5002/oasis/soa-rm-csbr.pdf Verificado em 09/07/2008.

[44] M. Papazoglou. *Service-oriented computing: Concepts, characteristics and directions*. Proceedings of the Fourth International Conference on Web Information Systems Engineering.IEEE Computer Society, p. 3-12, 2003, Washington, DC, USA.

[45] E. Gamma. *Design Patterns – Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[46] Microsoft Patterns and Practice - *Web Client Software Facoty* - http://www.codeplex.com/websf - Verificado em 17/08/2008

[47] Java Desing Patterns – Model-View-Controller - http://java.sun.com/blueprints/patterns/MVC.html - Verificado em 17/08/2008

[48] M. Endrei, J. Ang, A. Arsanjani, S. Chua, P. Comte, P. Krogdahl, M. Luo, T. Newling. *Patterns – Service-Oriented Architecture and Web Services*.IBM Red Book, 2004.

[49] Microsoft Corporation. *Configuration Management Application Block Overview*, http://www.msdnbrasil.com.br/Tecnologias/arquitetura/nova/blocos_disp/ger_config/cmab.htm - Verificado em 31/06/2008

[50] *IEEE Std 1471-2000 IEEE Recommended Practice for Architectural Description of Software-Intensive Systems – Description.* http://standards.ieee.org/reading/ieee/std_public/description/se/1471-2000_desc.html - Verificado em 14/05/2008.

[51] *OASIS UDDI Specifications.*http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm - Verificado em 10/08/2008

[52] *Microsoft .NET Framework 3 Community* - http://netfx3.com/content/WFHome.aspx - Verificado em 12/08/2008

[53] *XAML Overview* - http://msdn.microsoft.com/en-us/library/ms752059.aspx - Verificado em 12/08/2008

[54] OASIS - *OASIS Web Services Business Process Execution Language (WSBPELL) TC* - http://www.oasis-open.org/committees/wsbpel/ - Verificado em 12/08/2008

[55] Web Services Interoperability Organization – *Basic Profile 1.1*- http://www.ws-i.org/Profiles/BasicProfile-1.1.html - Verificado em 12/08/2008

[56] WS-I Organization´s Web Site - http://www.ws-i.org – Verificado em 12/08/2008

[57] W3C - *Extensible Markup Language (XML) 1.0 (Fourth Edition)* - http://www.w3.org/TR/REC-xml - Verificado em 12/08/2008

[58] W3C - *Simple Object Access Protocol 1.1* - http://www.w3.org/TR/2000/NOTE-SOAP-20000508/ - Verificado em 12/08/2008

[59] W3C – *Web Services Description Language 1.1 - http://www.w3.org/TR/2001/NOTE-wsdl-20010315* - Verificado em 12/08/2008

[60] W3C – *Hypertext Transfer Protocol 1.1* - http://www.w3.org/Protocols/rfc2616/rfc2616.html - Verificado em 12/08/2008

[61] W3C - *XML Schema* - http://www.w3.org/TR/2001/REC-xmlschema-1-20010502 - Verificado em 12/08/2008

[62] OASIS – *UDDI Version 2.0.3 Data Structure Reference* - http://uddi.org/pubs/DataStructure-V2.03-Published-20020719.htm - Verificado em 12/08/2008

[63] OASIS – *UDDI Version 2.0.4 API Specification - http://uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.htm* - Verificado em 12/08/2008

[64] Microsoft Patterns & Practices – *Web Service Software Factory Community* - http://www.codeplex.com/servicefactory - Verificado em 15/08/2008

[65] Microsoft Patterns & Practices – *Smart Client Software Factory Community* - http://www.codeplex.com/smartclient - Verificado em 15/08/2008

[66] Microsoft Patterns & Practices Developer Center – *Mobile Client Software Factory* - *http://msdn.microsoft.com/en-us/library/aa480471.aspx* - Verificado em 15/08/2008

[67] K. Pister, J. Kahn, B Boser. Smart Dust: *Wireless Sensor Networks of Millimeter-scale Sensor Nodes*. Eletronics Research Laboratory Research Summary, 1999.

[68] Computerworld – *Mobile & Wireless*. *Mighty motes for medicine, manufacturing, the military and more.*

http://www.computerworld.com/mobiletopics/mobile/story/0,10801,79572,00.html – Verificado em 19/07/2008

[69] Crossbow Technology.*Wireless Sensor Networks: Mica2*. http://www.xbow.com/Products/productsdetails.aspx?sid=174– Verificado em 19/07/2008

[70] Intel Research. Intel Mote: *Sensor Nets / RFID. http://www.intel.com/research/exploratory/motes.htm* - Verificado em 19/07/2008

[71] Crossbow Technology.*Wireless Sensor Networks: IMote2*. http://www.xbow.com/Products/productdetails.aspx?sid=253 – Verificado em 19/07/2008

[72] *Plataforma Stargate*. http://platformx.sourceforge.net/ - Verificado em 19/07/2008

[73] *Microsoft Silverlight*. http://www.microsoft.com/SILVERLIGHT - Verificado em 20/07/2008

[74] W3C - *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition) – W3C Recommendation*.http://www.w3.org/TR/soap12-part1/ - Verificado em 01/08/2008.

[75] Remote Method Invocation Home http://java.sun.com/javase/technologies/core/basic/rmi/index.jsp - Verificado em 15/08/2008

[76] J. Brassil. *Using Mobile Communications to Assert Privacy from Video Surveillance*. Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium.p. 290-301, 2005, Colorado, USA.

[77] HP LoadRunner Software - https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-11-126-17^8_4000_100__ - Verificado em 12/08/2008

[78] E. Zeeb, A. Bobek, H. Bonn, F. Golatowski.*Leassons Learned from implementing the Devices Profile for Web Services*. Proceedings of the Digital EcoSystems and Technologies Conference.p. 229-232, 2007, Cairms, Australia.