Uma Infraestrutura Baseada em Componentes para Desenvolvimento de Aplicações Pervasivas para Cuidados com a Saúde

Walter Onofre Guerra Filho

Dissertação de Mestrado submetida à Coordenação do Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande - Campus de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Processamento da Informação

Angelo Perkusich, D.Sc.
Orientador

Campina Grande, Paraíba, Brasil

Uma Infraestrutura Baseada em Componentes para Desenvolvimento de Aplicações Pervasivas para Cuidados com a Saúde

Walter Onofre Guerra Filho

Dissertação de Mestrado apresentada em Janeiro de 2010

Angelo Perkusich, D.Sc.
Orientador

Antonio Marcus Nogueira Lima, D.Sc.
Componente da Banca
Hyggo Almeida, D. Sc.
Componente da Banca

Campina Grande, Paraíba, Brasil, Janeiro de 2010

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL DA UFCG

G934i
2010

Guerra Filho, Walter Onofre.
Uma infraestrutura baseada em componentes para desenvolvimento de aplicações pervasivas para cuidados com a saúde /Walter Onofre Guerra Filho. — Campina Grande, 2010.
39 f.: il.

Dissertação (Mestrado em Engenharia Elétrica) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática.
Orientador: Prof. Dr. Angelo Perkusich.
Referências.

1. Arcabouço. 2. Computação Pervasiva. 3. Saúde - Cuidados. I. Título.

CDU 004.4'2(043)

UMA INFRAESTRUTURA BASEADA EM COMPONENTES PARA DESENVOLVIMENTO DE APLICAÇÕES PERVASIVAS PARA CUIDADOS DE SAÚDE

WALTER ONOFRE GUERRA FILHO

Dissertação Aprovada em 17.03.2010

ÂNGELO PERKUSICH, D.Sc., UFCG
Orientador

ANTONIO MARCUS NOGUEIRA LIMA, Dr., UFCG
Componente da Banca

HYGGO OLIVEIRA DE ALMEIDA, D.Sc., UFCG
Componente da Banca

CAMPINA GRANDE - PB
MARÇO - 2010

Resumo

Atualmente, estima-se que até 72% das mortes ocorridas por causas naturais sejam causadas por doenças crônicas. Estas doenças têm evolução lenta e contínua, e seu tratamento pode durar muitos anos. Em geral, este longo tratamento exige um acompanhamento constante do paciente para ser eficaz. O problema é que o custo deste tipo de tratamento é muito alto.

A forma atualmente mais aceita para tentar reduzir estes custos é alterar o paradigma de tratamento atual. Hoje, o paciente é visto como um receptor passivo do tratamento, enquanto neste novo modelo, ele é visto como um importante agente no cuidado de sua própria saúde. Este novo modelo é denominada autogerenciamento.

Para que haja o autogerenciamento, o paciente deve usar os sensores recomendados pelos médicos e um dispositivo móvel para capturar estes dados e mostrá-los ao paciente, bem como os enviar ao centro médico.

Este trabalho propõe uma infraestrutura baseada em componentes para facilitar a comunicação entre os sensores e as aplicações de nível de usuário do dispositivo móvel.

Abstract

Nowadays, there are estimates that 72% of all deaths are because of chronic diseases. These diseases have a slow and continuos progression, and it's treatment can take years. In most cases, the treatment requires a constant monitoring of the patient to be effective. The problem is that the costs of this kind of treatment are very high.

The most accepted way to reduce theses costs is to change the care taking paradigm. Now, patient is seen as a passive receptor of the treatment, while in the new paradigm, he is seen as an important agent for he's own health. This new paradigm is called self-care.

To implement this self-care paradigm, the patient must use the recommended sensors and a mobile device. This mobile device retrives the sensor's information and show them to the patient, at the same time that sends it to the medical center.

In this dissertation, a component-based infrastructure is proposed to make the communication between the sensors and the user-level applications on the mobile device easier.

Agradecimentos

À minha mãe, pelo amor incondicional em todos os momentos de minha vida.

Aos outros membros de minha família. Todos, em alguma medida, me ajudaram a chegar até aqui.

Ao meu orientador, Professor Angelo Perkusich, sem o qual este trabalho não existiria.

Ao Professor Hyggo Almeida, que teve a ingrata tarefa de ler os esboços que eu escrevia.

Ao Professor Antonio Marcus Nogueira Lima, que, embora não diretamente envolvido neste trabalho, me ensinou muito durante toda a minha vida acadêmica.

Aos amigos do Laboratório Embedded.

Aos grandes amigos da Signove, pelos recorrentes incentivos à conclusão do mestrado.

Aos professores e funcionários do DEE/COPELE.

À minha noiva Talita, pelo incentivo e apoio permanentes.

Lista de Figuras

Sumário

Capítulo 1

Introdução

Hoje em dia é crescente a preocupação com o bem-estar da população. Segundo a *Continua Health Alliance* [5], entre 60% e 85% da população mundial não faz exercícios físicos suficientes. Como resultado, essas pessoas têm mais propensão a ganho de peso e à contração de doenças crônicas, como hipertensão, diabetes e derrame. Ainda segundo a instituição, mais de 860 milhões de pessoas possuem pelo menos uma doença crônica e estima-se que este número chegue a três quartos da população mundial até 2020.

Esses dados são confirmados pela OMS (Organização Mundial de Saúde) que, em um relatório publicado em 2002 [6], estima que, em 2020, 80% de todas as doenças devem advir de problemas crônicos.

No Brasil, os dados são ainda mais preocupantes. Segundo a OMS, em 2005, foi estimado que 72% de todas as mortes, em todas as idades, foram causadas por doenças crônicas [7]. Um diagrama com o percentual de mortes por cada um dos tipo de doença é ilustrado na Figura 1.1.

Ainda segundo o mesmo documento da OMS, as projeções de morte para os próximos 10 anos no Brasil, são:

- Mais de 10 milhões de pessoas vão morrer por causa de doenças crônicas;
- Mortes por doenças infecciosas, de recém-nascidos e causadas por deficiência nutricional cairão 22%;
- Mortes por doenças crônicas subirão 22%, mais proeminentemente mortes por diabetes, que subirá 82%.

Para tentar diminuir estes números, a OMS sugere uma profunda alteração nos sistemas de saúde [6]. Segundo ela, o sistema de saúde atual não é eficaz no tratamento dessas

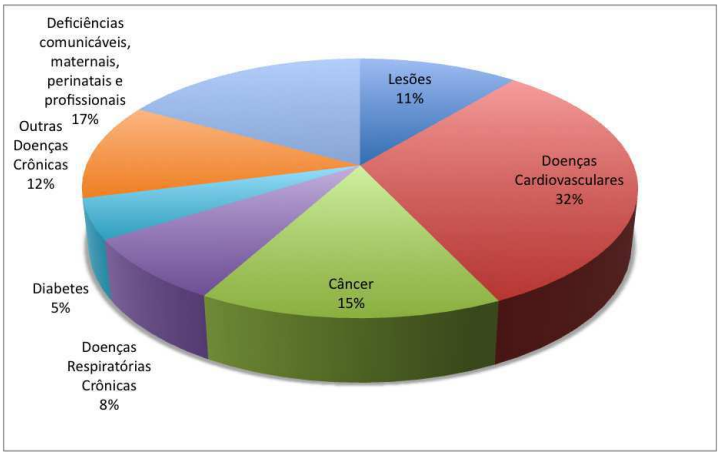

Figura 1.1: Projeção de mortes em 2005, por causa, em todas as idades

doenças pois se baseia no modelo de tratamento de casos agudos e episódicos. Como as doenças crônicas são problemas de longo prazo, exigem um contato regular com o paciente por um período de tempo extenso, frequentemente de anos. Além disso, as informações sobre o paciente devem ser centralizadas, de forma que múltiplos profissionais possam ter acesso às mesmas, como enfermeiros, preparadores físicos, nutricionistas e médicos.

Uma outra mudança sugerida é que o tratamento tenha como foco o paciente, sendo ele próprio mais responsável por sua saúde. Isto difere do modelo atual, onde o paciente é visto como um recebedor passivo do tratamento. Neste novo paradigma, chamado de autogerenciamento, o paciente leva uma vida mais independente, pois ele próprio tem informações sobre como deve agir para melhorar seu estado de saúde.

Além do aumento do número de pacientes com doenças crônicas, um outro dado preocupante encontrado pela *Continua Health Alliance* é a diminuição do número de profissionais na área de saúde, como médicos, enfermeiros e técnicos de suporte.

Atualmente, inúmeras pesquisas apontam que uma forma de amenizar este problema é ampliar a utilização de Sistemas de Auxílio ao Cuidado com a Saúde. Este tipo de sistema aumenta a eficiência do tratamento e reduz drasticamente os custos de operacionalização de uma monitoração constante do paciente [8].

Os sistemas de auxílio aos cuidados com a saúde são alvo de inúmeras pesquisas, e fazem parte de uma área denominada *Pervasive Healthcare*. São muitos os desafios inerentes ao desenvolvimento deste tipo de sistema, como o desenvolvimento de sensores não-intrusivos, o projeto de uma interface com o usuário fácil e intuitiva, pois quem deve implantar e operar estes sistemas são os profissionais de saúde, além do desenvolvimento de algoritmos de processamento dos dados dos sensores para obter informações de mais alto nível e auxiliar

médicos ou outros profissionais.

1.1 Problemática

Num típico sistema de *Pervasive Healthcare* existem quatro etapas fundamentais:

1. Acessar os dados dos sensores;
2. Disponibilizar os dados para algoritmos de processamento;
3. Processar estes dados de forma a obter informações relevantes e de mais alto nível;
4. Disponibilizar essas informações para fácil acesso pelos profissionais de saúde e pelo próprio paciente.

Um dos maiores problemas para o desenvolvimento deste tipo de software está nos items 1 e 2. pois não há uma forma padronizada de acesso aos dados de sensores como ECG (Eletrocardiograma), oxímetro, entre outros. Cada fabricante define o seu próprio protocolo de acesso e comunicação. Isto leva a sistemas como o *LifeShirt System* (VivoMetrics) e o *Philips TeleHealth* (Philips Medical Systems), que são vendidos como sistemas completos e fechados, mas não permitem interoperabilidade entre múltiplos fabricantes.

Atualmente, o padrão mais maduro em utilização é o ISO/IEEE 11073 [9]. Este padrão foi inicialmente concebido para a utilização em equipamentos médicos em UTIs. Isto leva a alguns problemas como a suposição de que sempre há rede disponível, o transporte de dados previsto é somente serial (RS-232) e não há preocupação com consumo de energia, pois os dispositivos estão sempre ligados à rede elétrica. Estas características o tornam impróprios para a utilização num sistema de auxílio ao cuidado com a saúde pervasivo. Uma extensão do padrão ISO/IEEE 11073 foi desenvolvida para adaptar-se a novos requisitos, como comunicação utilizando *Bluetooth* e perfis de baixo consumo de energia, para dispositivos movidos à bateria. Embora este novo padrão ainda esteja em estado de rascunho, ele já está sendo gradualmente adotado pelos fabricantes de dispositivos [10].

Com a crescente adoção do padrão, a interação entre os sensores está sendo facilitada. Entretanto, implementar o padrão e desenvolver aplicações que o utilizem ainda é um processo complexo e que requer um grande esforço [9]. Isto aumenta significativamente o tempo e o custo de desenvolvimento, uma vez que cada software tem que elaborar e desenvolver toda a esturura para coleta e tratamento dos dados dos sensores. Além disso, o custo de evolução do software também aumenta, pois dependendo da forma como o desenvolvedor faz o acesso aos dados, adicionar sensores pode ser muito complicado.

1.2 Objetivo

O objetivo deste trabalho é prover uma infraesturura para o desenvolvimento de aplicações pervasivas para Cuidados com a Saúde. A infraestrutura deve prover uma camada de abstração entre os sensores e a aplicação de forma que a adição, remoção e substituição dos sensores seja feita seguindo o modelo *plug and play*.

Utilizando as etapas enumeradas na sessão anterior de um típico sistema de auxílio ao cuidado com a saúde, este trabalho foca em facilitar os items 1 e 2, tirando do desenvolvedor a responsabilidade de implementar o protocolo definido no padrão e o acesso a cada sensor individualmente.

Esta intraesturura possui alguns requisitos importantes, como:

- Tem com plataforma alvo dispositivos móveis. Isto significa consumir o mínimo de memória e ter processamento otimizado;
- É multiplataforma;
- Provê uma interface única para acesso aos dados de todos os sensores, independentes do seu tipo e fabricante;
- Provê um mecanismo de segurança, de forma que somente pessoas autorizadas tenham acesso às informações dos sensores.
- É extensível em relação ao tipo de comunicação com os sensores, como Bluetooth, ZigBee e Serial (RS-232).

Além desses requisitos, um outro ponto muito importante é que a infraestutura deve permitir a adição, remoção ou substituição dos sensores em tempo de execução, sem que para isto o sistema precise ser reiniciado. Este é um requisito fundamental devido à natureza dos dados que estão sendo tratados. Uma interrupção na leitura dos sensores pelo tempo de reinicialização do sistema pode implicar numa perda séria de informação.

A infraestrutura foi implementada utilizando o framework Qt [11] em C++. Ele foi escolhido por ser multiplataforma e, entre as plataformas suportadas, incluem-se Symbian/S60 e Maemo.

A validação da infraestrutura foi realizada através da construção de um aplicativo que recebe os dados dos sensores e exibe alarmes na tela de um dispositivo móvel. Esses alarmes são construídos baseados num sistema de regras, de forma que ele seja útil para monitorar tanto pacientes quanto atletas em treinamento.

1.3 Relevância

Usando a infraestrutura proposta, o tempo de desenvolvimento de uma aplicação pervasiva para cuidados com a saúde será potencialmente reduzido, pois grande parte dos esforços de implementação serão focados nas regras de negócio da própria aplicação. Dessa forma, será reduzido significativamente o custo de desenvolvimento. Além disso, o desenvolvimento de componentes para novos dispositivos é facilitado por existir um esqueleto de componente pré-definido.

Uma outra vantagem é que as aplicações que utilizarem a infraestrutura proposta poderão evoluir em tempo de execução, reduzindo o custo de manutenção e evitando perda de dados importantes.

Por fim, a infraestrutura possibilita uma maior adequação dos sistemas de cuidado com a saúde aos princípios da Computação Pervasiva, uma vez que encapsula os procedimentos de configuração e acesso aos dados provenientes dos sensores sem a intervenção do usuário.

Este trabalho vem complementar os estudos no Laboratório de Sistemas Embarcados e Computação Pervasiva, abrindo uma nova frente de trabalho. Esta nova frente vai permitir a capacitação de alunos e desenvolvimento de pesquisa na área de monitoramento de pacientes e cuidados com a saúde.

1.4 Estrutura da Dissertação

O restante deste trabalho está organizado como descrito a seguir:

- No Capítulo 2, é apresentada a fundamentação teórica sobre temas necessários ao entendimento deste trabalho. Em sequência, são apresentados os seguintes temas: Sistemas Pervasivos, Sistemas de auxílio ao Bem-Estar Pervasivos e Desenvolvimento de sistemas orientados a componentes.

- No Capítulo 3 são descritos alguns trabalhos relacionados, apresentando comparações a abordagem proposta.

- No Capítulo 4 é descrita a infraestrutura proposta, detalhando seus componentes e a interação entre eles.

- No Capítulo 5, é descrita a forma de validação da estrutura proposta

- Por fim, no Capítulo 6 são apresentadas as conclusões do trabalho, apresentando as suas contribuições e discussões sobre trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Neste capítulo são apresentados os conceitos relativos a computação pervasiva, *pervasive healthcare* e ao modelo de sinais/*slots* implementado pelo *framework* Qt. Estes conceitos são importantes para o entendimento do restante do documento.

2.1 Computação Pervasiva

A Computação Pervasiva foi um conceito inicialmente proposto por Mark Weiser em 1991 [12]. Em sua visão, Weiser prega, de forma geral, um mundo no qual a computação está embutida nos objetos do dia-a-dia, como televisores, carros e roupas, estando os mesmos integrados às nossas vidas. Mais precisamente, tais objetos comunicam-se transparentemente uns com os outros, para nos apresentar informações e recursos a qualquer hora e em qualquer lugar, de acordo com nossas necessidades e preferências.

Weiser afirmou que a computação pervasiva poderia ser alcançada através de três elementos básicos: dispositivos baratos e com baixo consumo de energia, infraestrutura de rede sem fio, para permitir a comunicação entre estes dispositivos, e aplicações pervasivas. Na época desta afirmaçãoo, a tecnologia de hardware não estava preparada para suportar o paradigma da computação pervasiva. As redes sem fio, como as que temos atualmente, ou não estavam disponíveis ou não se encontravam embutidas nos dispositivos móveis. Como consequência, aplicações pervasivas não podiam ser desenvolvidas.

Entretanto, nos últimos anos este cenário começou a mudar, com o introdução de dispositivos móveis mais poderosos em termos de capacidade de memória e processamento e conectividade de rede. Porém, existem problemas associados à criação de sistemas de software para ambientes pervasivos como, por exemplo, a necessidade de acesso completo a informações de usuários e do ambiente em que o sistema está situado que nem sempre estão

disponíveis ou, quando estão disponíveis, precisam ser interpretadas e utilizadas de forma correta. Além disso, as mudanças no ambiente devem ser consideradas pelo sistema.

Outro aspecto fundamental da computação pervasiva é a invisibilidade [13]. Os sistemas pervasivos devem requerer o mínimo de intervenção do usuário, de forma que se aproximem da invisibilidade.

Há uma crescente migração da computação tradicional baseada em computadores pessoais para a computaçãoo pervasiva [14]. Sendo assim, existem vários projetos nessa área. Entre eles se destacam o projeto Aura [15] da Carnegie Mellon University, o projeto Endeavour, da University of California e o projeto Oxygen, do Massachusetts Institute of Technology. Esses trabalhos comprovam o constante crescimento do estudo e da utilização da computação pervasiva como solução de problemas nas diversas áreas como, por exemplo, problemas na área da medicina, do entretenimento ou da indústria dos transportes.

2.2 *Pervasive Healthcare*

A aplicação da computação pervasiva na área de medicina é chamada de a *pervasive healthcare*. O objetivo do *pervasive healthcare* é possibilitar a aplicação do modelo pervasivo proposto por Weiser para auxiliar nos cuidados com a saúde, como o suporte à vida independente, ao bem-estar (*wellness*) e ao auxílio ao tratamento de doenças [16].

Existem muitas vantagens na utilização de sistemas de *pervasive healthcare*, por exemplo:

Redução de custos Com a descentralização do tratamento, menos leitos são ocupados em hospitais, o que gera uma grande redução nos custos;

Aumento na qualidade do serviço Dispositivos pervasivos podem dar aos médicos acesso em tempo real ao histórico médico do paciente. Esta informação pode ajudá-los a tomar melhores decisões sobre os tratamentos. Além disso, sistemas de apoio ao diagnóstico podem reduzir os erros médicos;

Aumento da eficiência dos enfermeiros Como os sensores utilizados pelos pacientes os monitoram constantemente, os enfermeiros podem espaçar um pouco mais as visitas e ainda assim ter confiança no estado atual do paciente;

Como dito anteriormente, este paradigma de cuidados com a saúde é descentralizado e o paciente é considerado o principal agente de saúde, e não mais um mero recebedor passivo do tratamento. Neste modelo, o paciente utiliza sensores para monitorar os sinais vitais requisitados pelo médico e um dispositivo móvel que captura estes sinais. Ele também

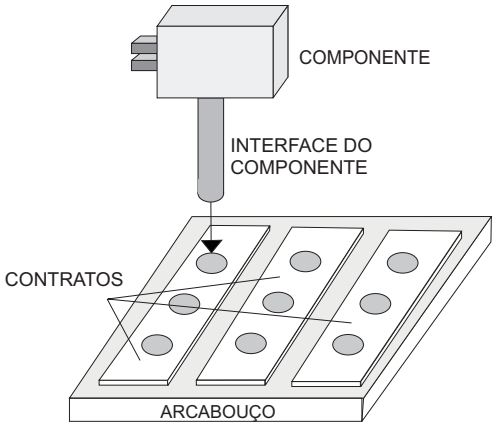

Figura 2.1: Arquitetura de componentes [1]

recebe um treinamento básico de como agir caso algum evento anormal ocorra. Além disso, o próprio médico pode requisitar o estado atual do paciente, que lhe é enviado pelo dispositivo móvel.

Estes conceitos são primordiais para o entendimento deste trabalho, pois ele tem como objetivo facitar o desenvolvimento deste tipo de aplicação.

2.3 Desenvolvimento de software baseado em componentes

O Desenvolvimento Baseado em Componentes (DBC) [17] tem obtido bastante espaço na produção de software, promovento aumento de produtividade e qualidade, em virtude da reutilização de componentes anteriormente implementados. Um componente de software é um artefato de software que obedece a um modelo de componentes e pode ser implantado e composto de forma independente, sem modificação, de acordo com um padrão de composição [18]. As abordagens de DBC geralmente são baseadas na arquitetura ilustrada na Figura 2.1, cujos elementos são descritos a seguir.

A *interface de componente* de software é representada pelas assinaturas dos métodos que implementam a funcionalidade do componente. O *arcabouço* de componentes é a infraestrutura que possibilita a "montagem" dos componentes para a construção de aplicações, coordenando a interação entre os componentes. Para garantir que os componentes se comportarão da maneira esperada pelo arcabouço, são definidos os *contratos*. Os contratos representam interfaces que o componente é obrigado a implementar. Eles garantem que o desenvolvimento de componentes independentes obedeça determinados padrões, fazendo

com que o arcabouço seja capaz de utilizá-los sem conhecer detalhes internos de implementação [19].

A separação entre a interface do componente e a sua implementação flexibiliza o desenvolvimento de software, uma vez que possibilita a reutilização de componentes criados por terceiros, baseando-se apenas nas suas características externas. Através dessas características, ou seja, da interface do componente, é realizado o processo de composição de componentes: a aplicação baseada em componentes é projetada com base na ligação entre as interfaces dos componentes.

Após o processo de montagem, os componentes são configurados de acordo com as características específicas da aplicação em questão. Nesta etapa de configuração são definidas, por exemplo, a URL de conexão de um componente de banco de dados ou a porta utilizada por um componente de comunicação. Essas informações não são conhecidas enquanto os componentes estão sendo desenvolvidos; portanto, são definidas apenas em uma etapa imediatamente anterior à execução da aplicação.

Uma vez que os componentes são compostos, configurados e integrados ao arcabouço, a aplicação pode então ser executada. Durante a execução da aplicação, os componentes interagem uns com os outros, de quatro formas diferentes: provendo serviços para outros componentes, acessando serviços de outros componentes, gerando eventos e observando eventos [20]. Durante o ciclo de vida de execução da aplicação, os componentes podem ser substituídos por versões otimizadas ou até mesmo por versões com correção de bugs. Mediante a alteração dos requisitos da aplicação, novos componentes também podem ser adicionados, acrescentando assim novas funcionalidades ao sistema.

Para que essas alterações nos componentes que constituem a aplicação possam ser realizadas sem interferir na execução do sistema, é necessário que o arcabouço que coordena a interação entre os componentes possibilite que a aplicação seja composta dinamicamente. Ou seja, mesmo durante a execução da aplicação, novos componentes podem ser adicionados, removidos ou alterados, permitindo que a aplicação possa evoluir de acordo com as mudanças de requisitos.

O foco na reutilização de artefatos de software previamente construídos, presente nas abordagens DBC, é uma característica importante para o desenvolvimento de aplicações no contexto de cuidados com a saúde. Uma vez que várias aplicações diferentes podem ser desenvolvidas para o mesmo domínio, é imporante reutilizar artefatos de software previamente implementados. Desta forma, promove-se um aumento da produtividade na concepção da aplicação. Pode-se ter também um aumento na qualidade da aplicação, visto que os componentes reutilizados já foram testados e validados em outras aplicações.

2.4 Modelo de Sinais/Slots de Qt

Um outro conceito importante para o entendimento do funcionamento da infraestrutura proposta neste trabalho é o modelo de sinais/slots providos pelo *framework* Qt. Toda a comunicação entre os componentes na infraestrutura desenvolvida utiliza este mecanismo.

Normalmente, a comunicação entre componentes em outros *toolkits* é feita utilizando o mecanismo de *callbacks*. Um *callback* é um ponteiro pra função, de forma que caso se deseje ser notificado de algum evento, o desenvolvedor passa esse *callback* para a função que irá notificá-lo. A função que realiza o processamento irá, então, manter um registro de todos os *callbacks* e irá chamá-los quando apropriado. Este mecanismo tem duas falhas fundamentais: eles não são seguros quanto ao tipo, não se pode garantir que a função de processamento chamará o *callback* com os argumentos corretos; o *callback* é fortemente acoplado com a função de processamento, já que ela precisa manter uma lista deles e executá-los.

Para resolver estes problemas, o *framework* Qt implementa um mecanismo denominado sinais/slots. Um *sinal* é emitido quando um determinado evento ocorre e um *slot* é uma função que é chamada em resposta a um sinal em particular. Este mecanismo resolve os dois problemas do mecanismo de *callback*:

- Existe segurança quanto ao tipo. A assinatura do sinal deve ser compatível com cada *slot*. Caso as assinaturas não sejam compatíveis o sistema não acopla o sinal ao slot, prevenindo erros durante a execução;
- Sinais e *slots* são fracamente acoplados. Uma classe que implementa um sinal não tem conhecimento de quais, se é que algum, *slot* é executado quanto o sinal é emitido.

Este mecanismo é implementado utilizando um sistema de pré-compilaçao, chamado *moc* (*meta-object compiler*).

Capítulo 3

Trabalhos Relacionados

Neste capítulos são mostrado alguns trabalhos que ajudam a resolver o problema do desenvolvimento de aplicações pervasivas de auxílio aos cuidados com a saúde. É feita, também, uma análise sobre vantagens e desvantagens de cada abordagem.

3.1 Wagner, 09 [3]

Este artigo trata do projeto OpenCare. Ele é um projeto e uma infraestrutura *Open Source* para implementação e teste de sistemas de auxílio ao dia-a-dia, uma área dentro do tema de *Pervasive Healthcare*.

Esta infraestrutura é direcionada a pesquisadores e empresas que desejam implementar o seu próprio sistema pervasivo de cuidados com a saúde *Open Source* sem ficarem presos a um único fabricante ou plataforma. Ao usar a plataforma proposta os pesquisadores ou empresas podem utilizar um grande número de sensores e dispositivos de diferentes fabricantes, além de desenvolver o seu próprio *hardware*.

O Projeto OpenCare é dividido em quatro níveis lógicos. Estes níveis lógicos, ilustrados na Figura 3.1, são:

Nível de Casa Consiste em sensores e atuadores presentes na casa do paciente. Geralmente consiste em uma tela *touch* fixa, ou mesmo um PC, incluindo um *framework* de agenda, de persistência de dados e de alame;

Nível Central Contém o servidor central e o banco de dados, assim como outros componentes da lógica de negócios, como histórico dos dados dos pacientes;

Nível Público Contém a interface para que os médicos, enfermeiros e todo a equipe necessária do hospital consiga monitorar o paciente. O Nível Público é disponibilizado

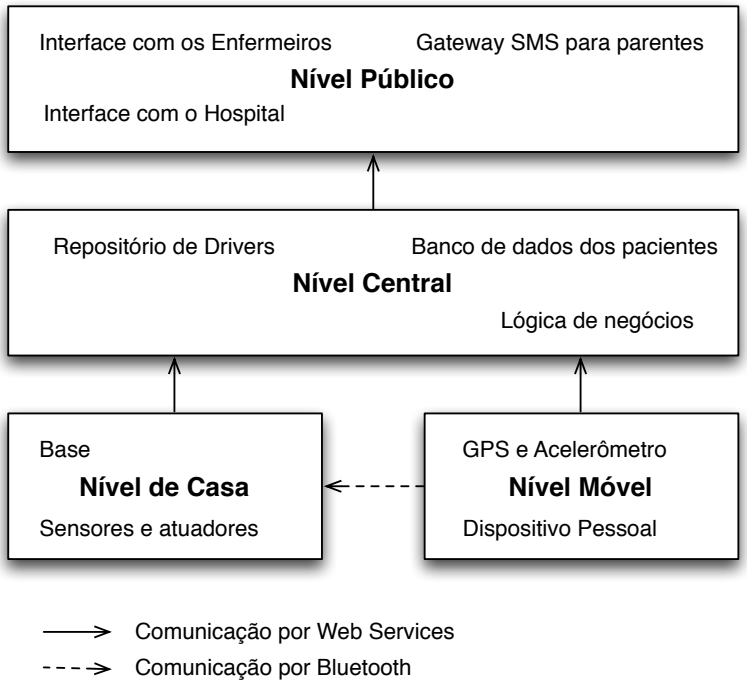

Figura 3.1: Os Níveis do Projeto OpenCare

através de Web Services e outros *middlewares* que permitam distribuição para sistemas heterogêneos;

Nível Móvel Ainda em desenvolvimento, este nível vai permitir que o paciente saia de casa e, ainda assim, continue sendo monitorado.

Como argumentado, uma das maiores métricas de sucesso de um sistema pervasivo de cuidados com a saúde é a usabilidade e eficiência. Não se pode esperar que parentes ou o pessoal de auxílio médico tenham conhecimentos técnicos suficientes para tratar de procedimentos complicados de instalação. Desta forma, todos os procedimentos de instalação devem ser feitos com o mínimo de interação com o usuário.

A infraestrutura suporta hoje em dia uma quantidade grande de tipos de sensores de diferentes fabricantes. Todos eles devem ser detectados automaticamente. O sistema também deve obter os *drivers* de dispositivo necessários de forma automática.

Uma deficiência do Projeto OpenCare é que ele não trata do caso onde existem vários pacientes na cujos alcances das bases se sobreponham. Dessa forma, ele não tem como garantir que um sensor vai ser pareado com a base do paciente correta.

3.2 EIHelw, 09 [4]

Neste artigo, o autor descreve um *framework* para sistemas pervasivos de cuidados com a saúde que utiliza um modelo arquitetural chamado de *push style message broker model*. Toda a comunicação no sistema é baseado em troca de mensagens assíncronas, componentes do sistema podem publicar dados em um tópico ou se inscrever para receber os dados de um tópico em particular ou de uma categoria de tópicos.

O sistema consiste de 3 Módulos:

Ambiente com Sensoriamento Pervasivo Neste módulo estão os sensores utilizados pelo paciente e sensores do ambiente. No paciente é utilizado um sensor na orelha que é capaz de definir parâmetros como batimentos cardíacos e temperatura do corpo e, para o ambiente, são usadas câmeras com FPGA imbutido, de forma que parte do processamento de vídeo seja feito localmente;

Fusão de Dados e Análise É neste módulo do sistema que é feita a inferência e aprendizado sobre os hábitos do paciente. São utilizados mecanismos de tomada de decisão probabilísticas e fusão de dados de baixo nível de sensores. Além disso, um padrão de comportamente é inferido utilizando cadeias escondidas de Markov;

Interface É a forma de interação tanto do paciente quanto da equipe médica de acompanhamento com o sistema de monitoramento. Esta interação de usuários com o sistema é feita através de Web Services.

Um ponto forte deste trabalho é o modelo arquitetural utilizado, pois ele permite um fraco acoplamento entre componentes. Entretanto, a infraestrutura proposta não é extensível, pois não utiliza nenhum padrão de comunicação aberto, de forma que caso um paciente resolva utilizar o sistema, ele ficará preso aos poucos sensores suportados pela infraestrutura. Além disso, o trabalho não trata do caso onde múltiplos pacientes estão num mesmo ambiente.

3.3 Zhu, 09 [2]

Este trabalho apresenta uma infraestrutura segura para um sistema autônomo de *body network* chamada Vesta. Os principais requisitos no desenvolvimento do Vesta foram:

Gerenciamento Autônomo O sistema deve ser auto-gerenciável com pouca intervenção ou configuração humana e deve ser adaptativo à mudanças no contexto;

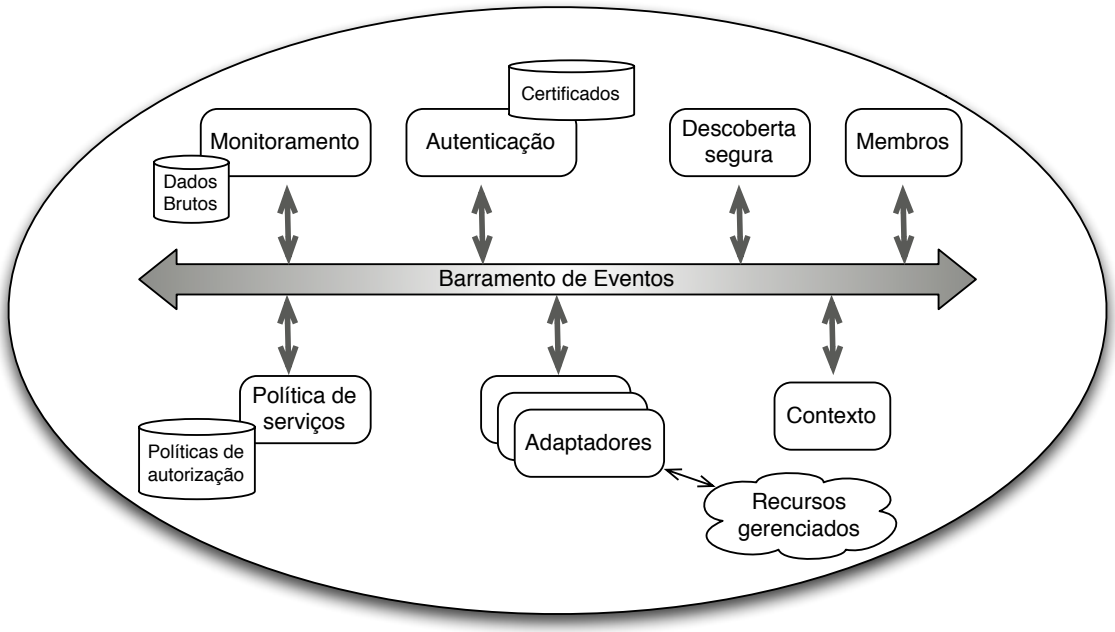

Figura 3.2: Arquitetura do Vesta [2]

Controle de Acesso Deve possuir níveis de acesso, de forma que somente usuários legítimos podem realizar intervencões autorizadas;

Confidencialidade dos Dados O sistema deve minimizar vulnerabilidades de segurança que possam comprometer os dados dos pacientes.

O Vesta utiliza uma arquitetura baseada no padrão SMC (*Single Managed Cell*) para representar uma de rede de monitoramento de um paciente. Uma visão geral da arquitetura utilizada é ilustrada na FiguraREF.

Na infraestrutura proposta, ilustrada na Figura 3.2, os sensores se comunicam com o sistema utilizando o padrão IEEE 802.15.4, enquanto a base, geralmente um *smartphone*, utiliza *bluetooth*. Para garantir que todos os módulos do sistema interajam utilizando uma infraestrutura unificada, toda a comunicação é realizada por um barramento de eventos único utilizando o mecanismo *publish/subscribe*.

A seguir serão descritos os principais módulos que compõe o Vesta:

Adaptadores São os *drivers* de dispositivo do sistema. São carregados sob demanda. Toda a comunicação entre os adaptadores e os sensores é feita de foram encriptada;

Descoberta Segura É o módulo responsável por descobrir novos sensores e serviços;

Membros Quando um novo sensor é adicionado ao sistema, este módulo monitora continuamente a sua presença, de forma que quando o sensor sair da rede, o adaptador correspondente é descarregado;

Política de serviços Este é o módulo responsável pela política de segurança do sistema. Ele utiliza um banco regras para definir, por operação, se um determinado usuário tem autorização para executá-la;

Monitoramento É onde são armazenados os dados de um paciente;

Contexto É responsável por lançar os eventos de mudança de contexto no barramento para que os outros módulos tomem conhecimento;

Autenticação É o módulo responsável pela autenticação de usuários e sensores. Para isto, este módulo utiliza um algoritmo baseado em PKI (*Public Key Infrastruture*).

A grande contribuição deste trabalho é a forma de autenticação de novos sensores e usuários ao sistema, de forma a garantir tanto a segurança dos dados dos pacientes quanto que um sensor é adicionado à rede do paciente correto. Entretanto, este trabalho não leva em consideração que os sensores podem utilizar diferentes tecnologias de comunicação. Isto limita bastante a quantidade de sensores que podem ser utilizados. Além disso, como o sistema foi pensado para um ambiente de hospital, ele não provê um mecanismo visualização dos dados dos sensores e interação pelo próprio paciente.

Capítulo 4

A Infraestrutura Proposta

Neste capítulo apresenta-se a infraestrutura proposta neste trabalho. São descritos os componentes que compõem o sistema, as suas respectivas interfaces e eventos e a interação entre eles para a realização das ações mais importantes.

A descrição é feita utilizando a nomenclatura do *framework* Qt. Nela, eventos são chamados de sinais e funções que capturam os eventos chamadas de *slots*.

4.1 Visão Geral

A infraestrutura proposta tem como objetivo facilitar o desenvolvimento de aplicações pervasivas para auxiliar no cuidado com a saúde e bem-estar. Para isto, ela faz a ponte entre os sensores e atuadores e as aplicações de usuário, sejam estas voltadas para dispositivos móveis, como um PDA ou *smartphone*, ou aplicações de grande porte, como um sistema de acompanhamento remoto de pacientes em um hospital.

Uma visão geral do sistema é ilustrada na Figura 4.1, onde são descritos os principais blocos que compõem um sistema de monitoramento de um paciente. São apresentados os sensores, como oxímetro, Balança, HRM (*Heart Rate Monitor*, ou Frequencímetro), que utilizam alguma tecnologia de comunicação sem fio para enviar o dados. Em seguida, tem-se a infraestrutura proposta, que se comunica com os sensores utilizando diferentes tecnologias e provê uma interface única de acesso aos dados. E, por fim, são descritas as aplicações que acessam a infraestrutura, como uma aplicação final para o usuário ou aplicações médicas de auxílio ao diagnóstico.

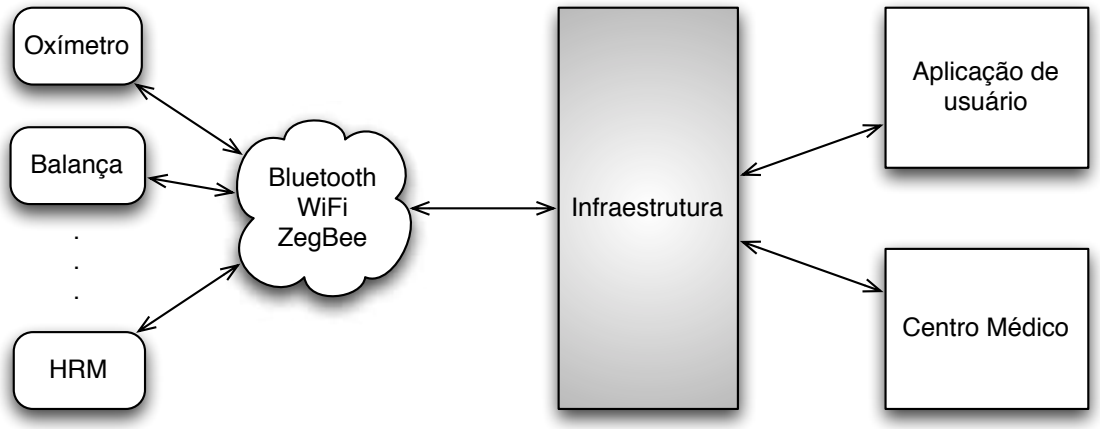

Figura 4.1: Visão Geral de um sistema de monitoramento

4.2 Componentes do sistema

Nesta sessão são descritos os componentes que compõe a infraestrutura. Serão explicitados a funcionalidade que cada um deve desempenhar e a sua interface.

4.2.1 Repositório de Componentes de Sensores

O "Repositório de Componentes de sensores", ou RCS, é um componente utilizado para procurar e carregar a partir da internet os componentes que implementam o tratamento de dados de novos sensores. Ele é utilizado toda vez que um novo sensor é detectado pelo sistema.

Para tentar obter o componente que trata os dados do novo sensor, o RCS recebe um identificador do sensor encontrado, que deve ser único. Esta busca pelo componente pode ser feita utilizando qualquer protocolo ou interface de rede.

Após encontrar o componente, o RCS faz o download do pacote e o repassa a outro componente para ser carregado e poder ser utilizado. A interface do RCS possui um só método:

```
boolean getSensorComponent(int sensorID)
```

Caso o RCS encontre o componente que trata o sensor, ele retorna TRUE, caso contrário, retorna FALSE.

4.2.2 Repositório de Componentes de Comunicação

O "Repositório de Componentes de Comunicação", ou RCC, é o componente utilizado para procurar e fazer o download de novos protocolos e interfaces de comunicação com sensores.

Este componente é executado periodicamente para descobrir se existem novos componentes de comunicação disponíveis. Este comportamento faz com que novos protocolos de comunicação possam ser adicionados ao sistema de forma automática, sem a intervenção do usuário.

Quando um novo componente de comunicação é encontrado, ele é carregado no sistema. A interface do RCC é descrita a seguir:

boolean findNewCommunicationComponents()

Este método é executado de forma assíncrona e retorna TRUE caso consiga se comunicar com a rede. O RCC emite um sinal quando um novo "Componente de Comunicação" é carregado com sucesso.

newCommunicationComponentLoaded()

4.2.3 Componente de Comunicação

Os "Componentes de Comunicação" encapsulam um protocolo específico de comunicação com os sensores. Estes componentes têm duas funções: fazer busca por novos sensores que utilizem o protocolo implementado e fazer a transferência de dados entre os componentes de sensores e os sensores físicos.

Uma camada extra para comunicação é necessária pois cada protocolo pode usar um formato diferente de pacote de comunicação. Então, caso não existisse esta camada, seriam necessários múltiplos componentes para tratar sensores do mesmo tipo, mas com protocolos de comunicação diferentes. Este componente elimina essa necessidade.

O Componente de Comunicação faz buscas periódicas por novos sensores. A interface do Componente de Comunicação é descrita a seguir:

boolean findNewSensors()
boolean sendData(Data data)

O método *findNewSensor* é executado de forma assíncrona e retorna TRUE caso consiga acessar a rede para fazer a busca de sensores, enquanto o método *sendData* é utilizado pelos "Componentes de Sensores" para enviar dados para os sensores.

Além destes métodos, o "Componente de Comunicação" emite os seguintes sinais:

newSensorFound(int sensorID)
sensorNotFound(int sensorID)
newDataAvailable(int sensorID, Data data)

O primeiro sinal é emitido quando um novo sensor é encontrado na rede. O segundo é emitido caso um "Componente de Sensor" tente se comunicar com um sensor indisponível. Já o terceiro sinal é emitido quando existe um novo dado disponível para algum sensor.

4.2.4 Componentes de Sensores

Os Componentes de Sensores tratam os dados recebidos por um, e somente um, tipo de sensor. Eles recebem os dados brutos de um "Componente de Comunicação", tratam os dados relevantes e os enviam para um componente que trata do armazenamento de dados.

A interface deste tipo de componente possui os seguintes métodos:

boolean configure()
boolean start()
boolean finalize()
void newDataAvailable(Data data)

O método *configure* é executado ao iniciar o componente de sensor ou quando o sistema é iniciado. Ele deve tentar estabelecer comunicação com o sensor físico e fazer as configurações necessárias. O segundo método implementado pelo "Componente de Sensor" é o *start*, que deve iniciar a captura de dados do sensor físico. O "Componente de Comunicação" também implementa um método *finalize*, que libera todos os recursos utilizados pelo componente. Este método deve ser executado antes do componente ser removido do sistema. Por último, o método *newDataAvailable* é executado quando um novo dado está disponível para o sensor.

4.2.5 Gerenciador de Sensores

O "Gerenciador de Sensores" é o componente responsável por carregar, descarregar e gerenciar os eventos enviados para os "Componentes de Sensores".

Este componente é necessário pois quando existe um novo dado disponível o "Componente de Comunicação" envia o mesmo evento independente de para qual o sensor o dado se destina, com o ID do sensor como parâmetro do evento. Dessa forma, uma das responsabilidades do "Gerenciador de Sensores" é receber este evento e repassar somente para o "Componente de Sensor" correto.

A Interface do "Gerenciador de Sensores" possui os seguintes métodos:

boolean loadSensor(CS sensor)
boolean unloadSensor(CS sensor)
boolean newDataAvailable(int sensorID, Data data)

O primeiro método é utilizado para carregar um "Componente de Sensor" no sistema. O método *unloadSensor* é usado para descarregar um "Componente de Sensor" do sistema e, por fim, o método *newDataAvailable* serve para capturar os eventos do "Componente de Comunicação" e o enviar ao "Componente de Sensor" correto.

O "Gerenciador de Sensores" envia um sinal:

sensorComponentLoaded(int sensorID)

Este sinal é enviado quando um novo "Componente de Sensor" é carregado.

4.2.6 Componente de Armazenamento de Dados

O "Componente de Armazenamento de Dados", ou CAD, é responsável por encapsular o banco de dados utilizado para armazenar os dados dos sensores. Ele armazena todo o histórico dos dados inseridos, com data e hora em que foi inserido no banco, o que é importante pois o médico ou outro profissional pode precisar de um intervalo de tempo grande para diagnosticar alguma anormalidade.

Como a quantidade de dados pode crescer muito com o passar do tempo, já que todo o histórico é armazenado, o CAD pode replicar o banco de dados local num remoto. Isto facilita a escalabilidade do sistema.

A interface deste componente possui os seguintes métodos:

boolean storeData(Data data)
Data retriveData(Time time = 0, Sensor sensor = 0)

O método *storeData* é chamado para incluir um dado novo no banco de dados. Quando este método é executado, o CAD verifica qual componente o executou e armazena junto com a informação.

Para acesso aos dados, existe o método *retriveData*, ele tem dois parâmetros opcionais que funcionam como filtros: o tempo inicial da consulta, de forma que se pode acessar somente os dados das últimas duas horas, por exemplo; e o sensor que inseriu os dados, para uma consulta dos dados de um só tipo de sensor.

4.2.7 Componente de Segurança

Em aplicações de *healthcare*, a confidencialidade dos dados e a privacidade do paciente é uma questão importante, principalmente quando estão involvidos políticos ou celebridades [2].

Além disso, controle de acesso também é uma necessidade. Em ambientes de hospital, por exemplo, enfermeiros e médicos precisam acessar dados dos pacientes e, frequentemente, executar operações no sistema. Também é necessário que o sistema possua diferentes privilégios de acesso a diferentes profissionais, pois uma ação incorreta pode resultar em sérias consequências.

Então, o Componente de Segurança tem basicamente dois problemas a resolver:

- Garantir que um sensor vai ser acessado pelo paciente correto. Isto é importante quando se tem vários pacientes num mesmo ambiente;
- Garantir que somente as pessoas autorizadas têm acesso aos dados desses sensores, incluindo diferentes privilégios para diferentes profissionais.

A partir dos requisitos expostos, foi definida a seguinte interface para o Componente de Segurança:

boolean autorizeNewSensor(int sensorID)
String authenticateUser(UserData data)
boolean checkUserAuthentication(String token)

O método *autorizeNewSensor* é executado sempre que um novo sensor é achado por um Componente de Comunicação. Este sensor autoriza o acesso ao sensor e a consequente captura de dados. Seguindo os princípios da computação pervasiva, a adição de novos sensores deve ser o menos intrusiva possível para o usuário. Inúmeros estudos [21] [22] já foram realizados e algumas das formas de interação sugeridas foram: reconhecimento de voz; leitores de códigos de barras; adição de um dispositivo touch-screen com teclado virtual; e, por último, NFC (*Near Field Communication*). Cada uma dessas formas tem vantagens e desvantagens, de forma que o desenvolvedor do sistema pode escolher qual sistema de interação será usado simplesmente implementando este componente.

Os outros dois métodos, *authenticateUser* e *checkUserAuthentication* são utilizados para garantir privilégios de acesso. Antes de iniciar a primeira operação, o usuário deve usar o método *authenticateUser*, que retorna uma *string* aleatória que serve como token de autenticação. A partir deste momento, todas as operações de leitura e escrita nos sensores e atuadores presentes no paciente devem ter como primeiro parâmetro o *token* provido. Desta forma, cada operação deve ser autenticada, o que garante diferentes privilégios de acesso.

4.2.8 Componente de Interface

O "Componente de Interface", ou CI, é responsável por toda a comunicação da infraestrutura com componentes externos. É ele quem processa as requisições da interface com o usuário ou de outro sistema que utilize a infraestrutura.

O CI deve ser a única forma de acesso externo à infraestrutura, de forma que alguns métodos providos por ele são simplesmente repassados para os componentes responsáveis. Além disso, o CI deve enviar eventos sobre novos sensores disponíveis para que a interface gráfica ou outros sistemas externos possam ser atualizados.

Os métodos providos pelo CI são:

String authenticateUser(UserData data) DataTypeArray getSensorData(String token, Time time = 0, int sensorID = 0)

O primeiro método simplesmente repassa a requisição do "Componente de Segurança", enquanto o segundo método verifica se o *token* tem permissão suficiente e, caso positivo, faz a requisição ao "Componente de Armazenamento de Dados".

O CI lança os seguintes eventos:

newSensorAvailable(int sensorID) systemRunning()

O primeiro evento é emitido quando um novo sensor é configurado e está recebendo dados. Já o segundo evento é emitido após a inicialização para indicar que ela ocorreu com sucesso.

4.3 Sequências de interação entre os Componentes para a realização de Ações

Nesta sessão são descritas as interações entre os diversos componentes do sistema para realizar as operações necessárias. São descritas as operações de inicialização, adição de um novo sensor, remoção de um sensor, leitura dos dados de um sensor e obtenção dos dados de um sensor por um usuário.

4.3.1 Inicialização

Os passos da inicialização do sistema, ilustrados na Figura 4.2, são descritos a seguir:

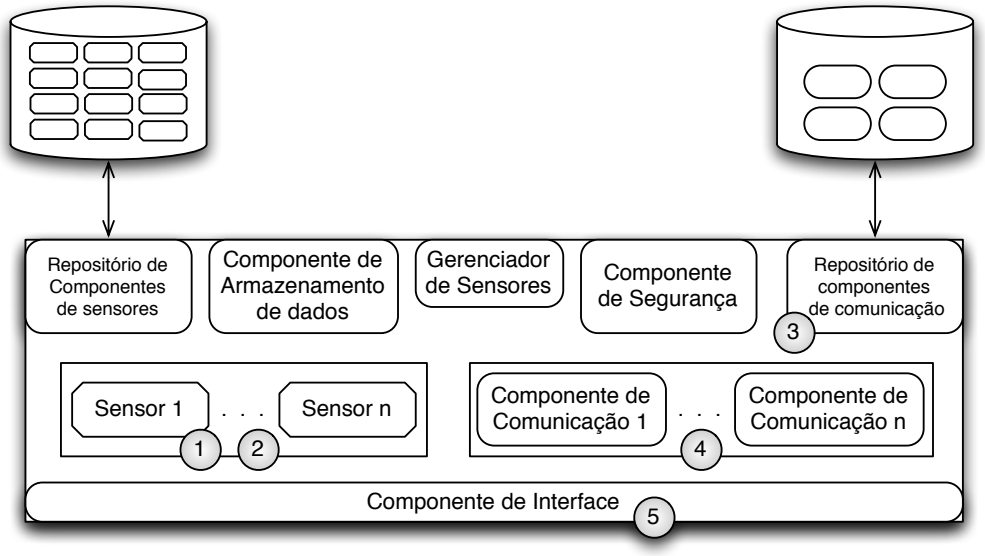

Figura 4.2: Sequência de inicialização do sistema

1. A primeira etapa da inicialização é executar o método *configure* em cada um dos sensores previamente configurados;

2. Em seguida, em cada um dos sensores cujo método *configure* retornou TRUE, é executado o método *start*;

3. A próxima etapa é executar o método *findNewCommunicationComponents* do "Repositório de Componentes de Comunicação". Este método é executado de forma assíncrona e procura por novos componentes de comunicação;

4. Então, para cada Componente de Comunicação instalado, é chamado o método *findNewSensors*, que é responsável por procurar novos sensores. Este método também é executado de forma assíncrona;

5. Por fim, o "Componente de Interface" emite um evento de *systemRunning* para indicar que a inicialização foi concluída com sucesso.

4.3.2 Adição de um novo sensor

O processo de adição de um novo sensor, ilustrado na Figura 4.3, é descrito a seguir:

1. Quando um novo sensor é encontrado, o Componente de Comunicação que o encontrou emite um sinal *newSensorFound*, com o ID do sensor como parâmetro;

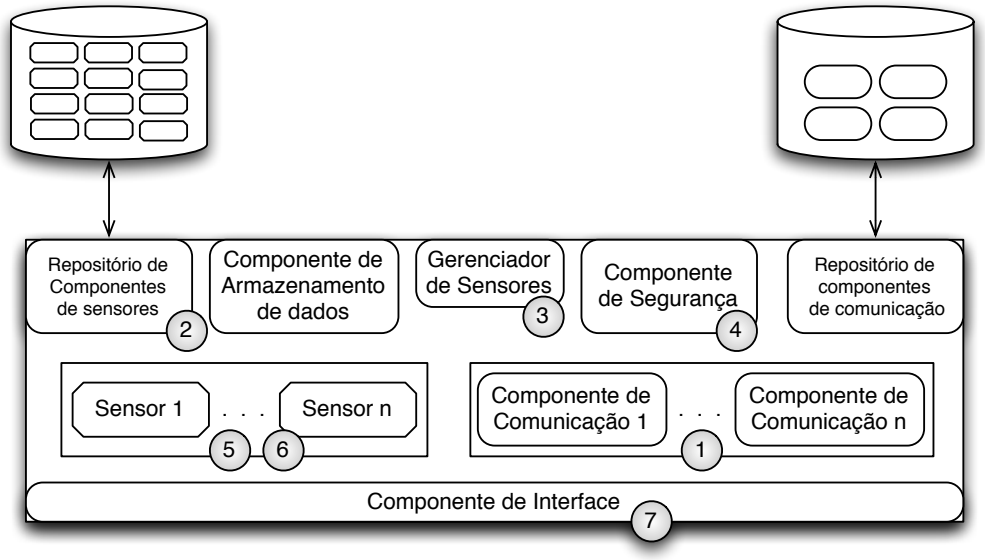

Figura 4.3: Sequência de adição de um novo sensor ao sistema

2. O "Repositório de Componentes de Sensores" recebe o sinal e começa a busca por um componente que trate os dados com sensor com o ID enviado;

3. Caso encontre o componente correto, o RCS faz o download do componente, e executa o método *loadSensor* do "Gerenciador de Sensores", que carrega o "Componente de Sensor" e emite um evento de *sensorComponentLoaded*;

4. O "Componente de Segurança e Autenticação" recebe este sinal e executa o seu método *authorizeNewSensor*. Como descrito na Seção 4.2.7, este método deve confirmar que este sensor pertence ao paciente correto;

5. Caso a autenticação seja efetuada com sucesso, o método *configure* do componente é executado;

6. Em seguida, caso o método *configure* retorne TRUE, é executado o método *start* do componente do sensor, que inicializa a coleta de dados

7. Por fim, o "Componente de Interface" emite um evento de *newSensorAvailable*, para que os sistemas externos possam ser atualizados.

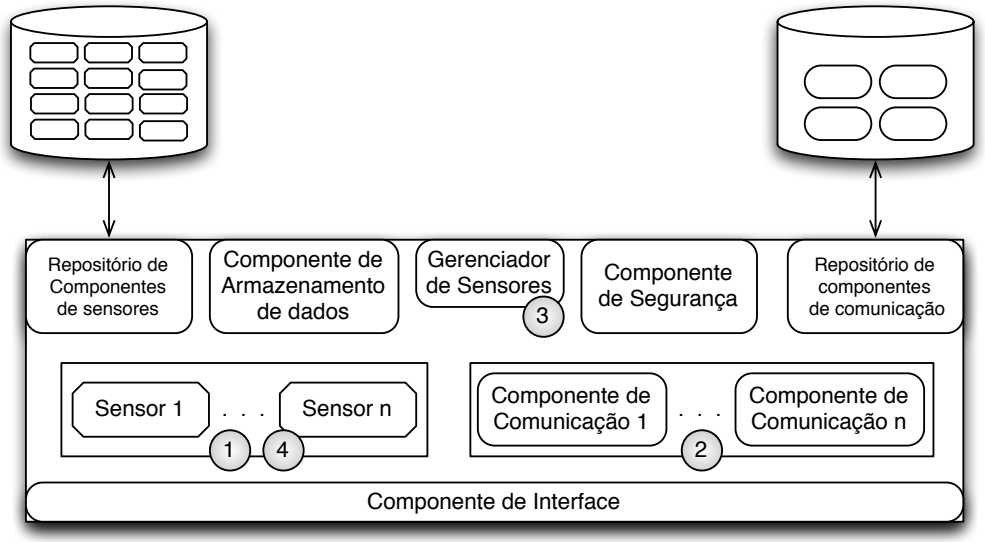

Figura 4.4: Sequência de remoção de um sensor do sistema

4.3.3 Remoção de um sensor

Quando um sensor configurado não pode ser encontrado pelo "Componente de Comunicação" ele é removido do sistema, de forma a reduzir o consumo de memória. O procedimento de remoção, ilustrado na Figura 4.4, é descrito a seguir:

1. Quando o sistema é inicializado, o primeiro método executado é o método *configure* de um dado "Componente de Sensor". Durante o *configure*, o CS tenta estabelecer comunicação com o sensor físico utilizando o "Componente de Comunicação";

2. Caso o sensor não seja encontrado, o "Componente de Comunicação" emite um sinal de *sensorNotFound*;

3. O "Gerenciador de sensores", então, captura este sinal e chama o método *finalize* do "Componente de sensor" correspondente;

4. Finalmente, o "Componente de Sensor" cujo método *finalize* foi executado é removido do sistema.

(Como fazer se um sensor já for removido e aparecer de novo? Pede confirmação de novo?)

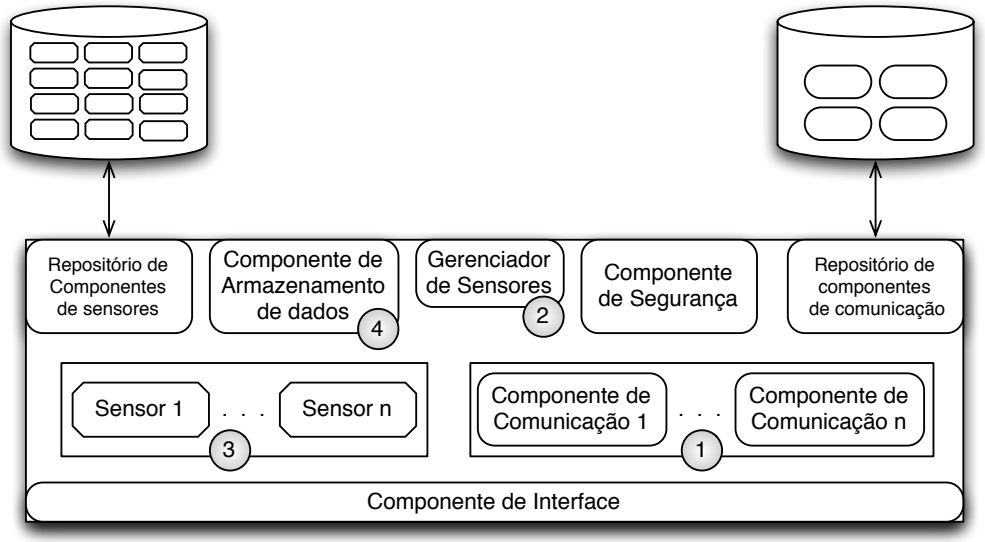

Figura 4.5: Sequência de leitura de um novo dado do sensor

4.3.4 Leitura dos dados de um sensor

O procedimento de leitura dos dados de um sensor, ilustrado na Figura 4.5, é descrito a seguir:

1. Quando um novo dado está disponível, o "Componente de Comunicação" o recebe, separa o dado enviado das camadas de transporte e lança um evento de *newDataAvailable*;

2. Este evento é capturado pelo "Gerenciador de Sensores" que, utilizando o sensorID enviado no evento, executa o método *newDataAvailable* do sensor correspondente;

3. O "Componente de Sensor" pode, então fazer o processamento desejado;

4. Em seguida, com as informações relevantes, o "Componente de Sensor" executa o método *storeData*, implementado pelo "Componente de Armazenamento de Dados";

4.3.5 Obtenção pelo usuário dos dados de um sensor

1. Antes de fazer a primeira consulta ao sistema, o método *authenticateUser* do "Componente de Interface" deve ser executado;

2. O "Componente de Interface" executa o método *authenticateUser* do "Componente de

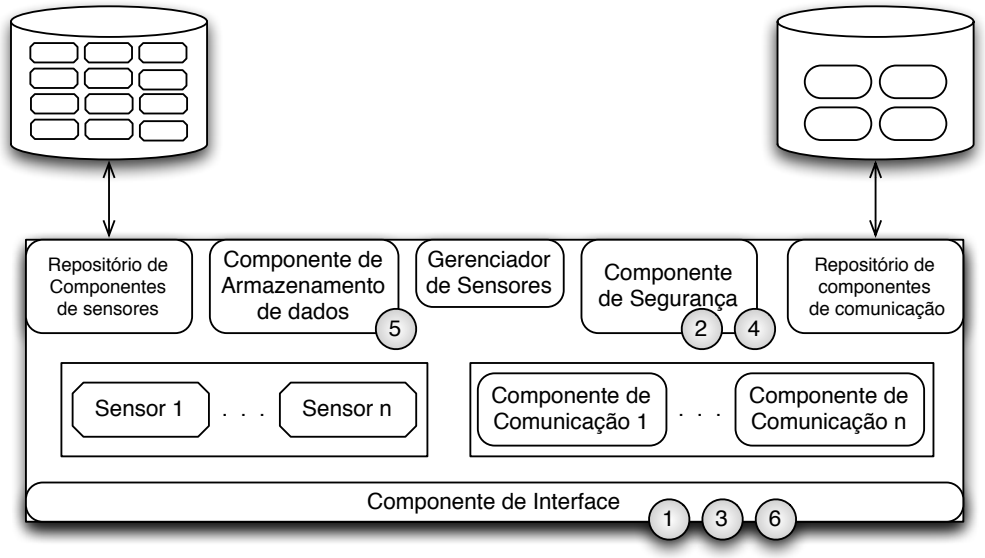

Figura 4.6: Sequência de obtenção dos dados de um sensor

Segurança", que verifica as permissões e retorna uma string de autenticação, chamada de Token. O Token, por sua vez, é retornado pelo "Componente de Interface";

3. De posse do token, o método *getSensorData*, do "Componente de Interface" deve ser executado. Chamadas subsequentes não necessitam do mesmo procedimento de autenticação;

4. O "Componente de Interface" executa o método *checkUserAuthentication* do "Compnente de Segurança" para garantir que aquele usuário tem permissão de acesso àquele dado;

5. O "Componente de Interface", então, executa o método *retriveData* do "Componente de Armazenamento de Dados", que retorna os dados solicitados;

6. Finalmente, o dado requisitado é retornado pelo "Componente de Interface".

A partir da utilização da infraestrutura proposta, o desenvolvedor de aplicações pervasivas para auxílio aos cuidados com a saúde pode se preocupar somente com as regras de negócio da própria aplicação. Isto elimina a necessidade deste desenvolvedor implementar um mecanismo de comunicação com os sensores. Além disso, a manutenção do software também é facilitada, pois a aplicação não precisa dar suporte a novos sensores.

No caso de novos tipos de sensores passarem a ser suportados pelo padrão ISO/IEEE 11073, a implementação destes novos componentes também é facilitada, pois existe uma interface bem definida que eles devem seguir e a camada de comunicação de baixo nível já está pronta. Portanto, a implementação de um componente que trate os dados de um novo tipo de sensor se resume ao tratamento dos campos de dados definidos no padrão.

Como o padrão é independente da camada do protocolo de transporte de dados, novos métodos de comunicação podem surgir. Neste caso, a adição do suporte a este novo protocolo deve ser feito implementando um novo "Componente de comunicação". Uma vez implementado este componente, todos os tipos de sensores previstos no padrãoo que utilizarem tal mecanismo de comunicação serão suportados nas aplicações.

Capítulo 5

Estudo de Caso

Para validar a infraestrutura proposta, é apresentado um sistema que a utilize. Este sistema proposto possui os seguintes módulos:

- Sensores;
- Dispositivo Móvel;
- Servidor Central;
- Terminal Médico.

Os *sensores* são responsáveis por capturar os dados e enviá-los ao restante do sistema usando uma tecnologia de rede de baixo alcance, como o bluetooth, por exemplo. O *dispositivo móvel* age como o terminal, configurando as conexões, processando os dados recebidos, inferindo estados médicos, e possivelmente mostrando-o ao usuário e enviando esta informação por uma tecnologia de rede de longo alcance, como GPRS ou 3G.

O *Servidor Central* provê os seguintes serviços:

Banco de dados de componentes Possui os componentes para cada tipo de sensor;

Banco de dados de tipos É um banco de dados gerado por um especialista com "estados médicos" reconhecidos, como: obeso, acima do peso, gripado, com arritmia, entre outros;

Banco de dados de monitoramentos Este banco de dados contém os monitoramentos recomendados por um médico para um determinado paciente.

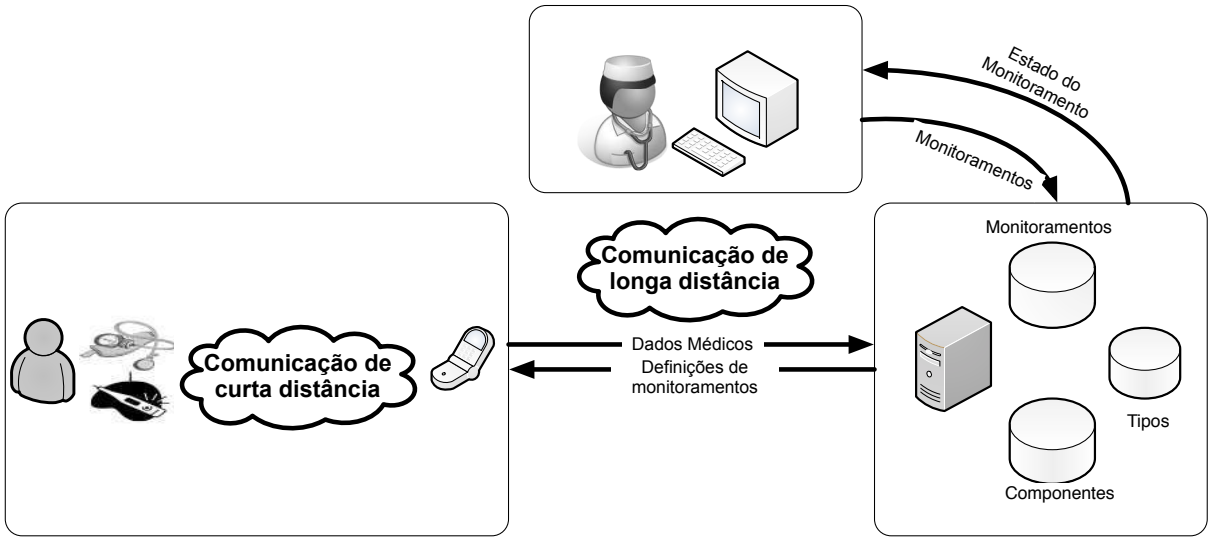

Figura 5.1: Visão Geral da aplicação

A utilidade e a forma de utilização destes bancos de dados é descrita adiante.

O *Terminal Médico* é o dispositivo utilizado pelo profissional de saúde para acompanhar o paciente e enviar monitoramentos recomendados. Uma visão geral da aplicação é mostrada na Figura 5.1.

O sistema deve operar da seguinte forma:

- O médico analisa o paciente e recomenda algum monitoramento específico, como o uso constante de um oxímetro ou o monitoramento diário do peso, por exemplo;
- Em seguida ele define um conjunto de regras, da forma: se *condição*, então *predicado*, onde *condição* é um tipo definido no *banco de dados de tipos* e *condição* é um alarme, um e-mail, um sms ou alguma outra saída suportada. Um exemplo de regra deste tipo é: se *gripado*, então envie um sms para número 1234567. Tanto nas condições quanto nos predicados, podem ser usados os operadores lógicos E, OU e NÃO;
- De posse dessas regras, o sistema procura no *banco de dados de componentes* quais possuem as saídas usadas nas condições. Estes componentes são, então, carregados no sistema;
- Agora, é analisada a entrada de dados destes componentes para determinar quais sensores estão presentes. Numa situação ideal, o paciente está utilizando todos os sensores recomendados pelo médico, de forma que todas as regras de monitoramento sejam ativadas. Entretanto, o paciente pode estar usando somente parte dos sensores.

Neste caso, o sistema tenta ativar o maior número de regras possíveis com os sensores disponíveis;

- Para cada regra ativada o *Servidor Central* é notificado e este, por sua vez, notifica o *Dispositivo Médico*. Isto é importante para que o médico acompanhe quais regras definidas por ele estão sendo monitoradas e possa ter um acompanhamento mais preciso do paciente.

No sistema definido, a infraestrutura é usada para prover a camada de abstração entre os sensores e a aplicação, de forma que o protocolo de comunicação com os sensores existentes e o mecanismo de busca e configuração de novos sensores foi completamente abstraído pelo desenvolvedor de aplicações. Além disso, caso novos sensores sejam desenvolvidos por outros fabricantes, a aplicação não precisa ser alterada dar suporte a estes novos sensores.

Capítulo 6

Conclusão e Trabalhos Futuros

O atual envelhecimento da população mundial, principalmente dos países desenvolvidos, requer que novos paradigmas de cuidados com a saúde sejam desenvolvidos. Cada vez mais o modelo de tratamento onde o paciente é também resposável por sua saúde ganha força, em detrimento do antigo paradigma, onde o paciente é um recebedor passivo do tratamento.

Segundo estudos, é bastante custoso manter um acompanhamento constante do paciente. Isto é ainda mais grave no caso de portadores de doenças crônicas, cujo tratamento pode durar anos ou até o fim da vida.

Para tentar reduzir estes custos, a atual aposta mais promissora é o desenvolvimento de sistemas de auxílio aos cuidados com a saúde. Estes sistemas tem como objetivo principal monitorar o paciente, em geral em sua própria casa, sem que seja necessário dedicar um profissional em tempo integral para isto. Além desta redução de custos, também aumenta-se a qualidade de vida do paciente, pois ele não perde a liberdade por ter um profissional sempre ao seu lado.

Com a evolução do paradigma da Computação Pervasiva, iniciou-se uma nova área de estudos, a dos sistemas pervasivos de cuidados com a saúde. Estes sistemas provêem o monitoramento desejado com a grande vantagem de requererem o mínimo de intervenção e configuração do usuário. Além disto, sistemas pervasivos não prendem o paciente à sua casa, pois tanto os sensores quanto a base de captura dos dados são portáveis o suficiente para não atrapalharem o dia-a-dia normal. Isto foi possível pela evolução de três grandes características dos dispositivos móveis atuais: a conectividade disponibilizada pelas tecnologias de rede sem fio, a mobilidade obtida com a redução do tamanho dos dispositivos e a capacidade computacional conseguida com os avanços no desenvolvimento de componentes eletrônicos, como processadores e memória.

Apesar da importância, ainda é bastante complexo desenvolver sistemas pervasivos de

auxílio aos cuidados com a saúde. Parte desta dificuldade vem da própria interdisciplinaridade da área. Além disto, a falta de um padrão amplamente adotado faz com que a reutilização de software e a interoperabilidade entre diferentes sistemas sejam praticamente nulos.

Para tentar resolver este problema, foi criado o padrão ISO/IEEE 11073, que procura padronizar o acesso aos sensores pelas bases de captura de dados. Apesar disto, para desenvolver um sistema completo, o desenvolvedor deve implementar o padrão e a aplicação que o utilize.

Mediante esta necessidade, este trabalho apresentou uma infraestrutura baseada em componentes para facilitar o desenvolvimento de aplicações pervasivas para os cuidados com a saúde. Esta infraestrutura disponibiliza um conjunto de componentes que permitem a reutilização das funcionalidades implementadas em diversas aplicações.

Além disso, a infraestrutura desenvolvida oferece suporte para a composição dinâmica de aplicações, possibilitando a adição, remoção e troca de componentes em tempo de execução. Por exemplo, o suporte a um novo tipo de sensor ou uma mudança na política de segurança dos dados podem ser implementados e adicionados à aplicação sem que para isto a aplicação precise ser reiniciada.

Como os dispositivos médicos não chegaram a tempo, são listados como trabalhos futuros testar a infraestrutura com dispositivos reais, de forma a montar um sistema completo para estudos e fazer uma análise precisa de desempenho em diferentes plataformas, como o Maemo, o Symbian/S60 e, possivelmente, o iPhone e o Android.

Referências Bibliograficas

[1] G. V. Ferreira, E. C. Loureiro Filho, W. F. A. Nogueira, A. F. A. Gomes, H. O. Almeida, and A. Frery. Uma Abordagem Baseada em Componentes para a Construção de Edifícios Virtuais . In *Proceedings of VII Symposium on Virtual Reality - SVR 2004*, volume 7, pages 279–290, São Paulo, 2004. Vida & Consciência.

[2] Yanmin Zhu, Morris Sloman, Emil Lupu, and Sye Keoh. Vesta: A secure and autonomic system for pervasive healthcare. *Pervasive Computing Technologies for Healthcare, 2009. PervasiveHealth 2009. 3rd International Conference on*, pages 1–8, 2009 2009.

[3] Stefan Wagner and Claus Nielsen. Opencare project: An open, flexible and easily extendible infrastructure for pervasive healthcare assisted living solutions. *Pervasive Computing Technologies for Healthcare, 2009. PervasiveHealth 2009. 3rd International Conference on*, pages 1 – 10, Apr 2009.

[4] M ElHelw, J Pansiot, D McIlwraith, R Ali, B Lo, and L Atallah. An integrated multi-sensing framework for pervasive healthcare monitoring. *Pervasive Computing Technologies for Healthcare, 2009. PervasiveHealth 2009. 3rd International Conference on*, pages 1 – 7, Apr 2009.

[5] Continua Health Alliance. http://www.continuaalliance.org/. Acessado em 10/12/2009.

[6] OMS. Cuidados inovadores para condições crônicas: Componentes estruturais de ação. relatório mundial. pages 1–105, May 2003.

[7] World Health Organization. The impact of chronic desease in brasil. pages 1–2, Jan 2009.

[8] T Laakko, J Leppanen, J Lahteenmaki, and A Nummiaho. Multipurpose mobile platform for telemedicine applications. pages 245 – 248, Tampere, Jan 2008.

[9] M Martinez-Espronceda. Implementing isodeee 11073: Proposal of two different strategic approaches. page 4, Vancouver, Jan 2009.

[10] M Clarke, D Bogia, and K Hassing. Developing a standard. *Proceedings of the 29th Annual International Conference of the IEEE EMBS*, page 3, Lyon, 2007.

[11] QT Framework. www.qt.nokia.com. Acessado em 10/12/2009.

[12] M Weiser. An integrated multi-sensing framework for pervasive healthcare monitoring. *Scientific American*, pages 94 – 104, 265(3) 1991.

[13] D. Saha and A. Mukherjee. Pervasive computing: A paradigm for the 21st century. *28th IEEE Conference on Local Computer Networks*, pages 25 – 30, 2003.

[14] H. Almeida E. Loureiro, G. Ferreira and A. Perkusich. Pervasive computing: What is it anyway? *Ubiquitous and Pervasive Knowledge and Learning Management: Semantics, Social Networking and New Media to their full potential*, pages 1 – 34, 2007.

[15] Daniel P. Siewiorek David Garlan and Peter Steenkiste. Project aura: Toward distraction-free pervasive computing. *IEEE Pervasive Computing*, pages 22 – 31, 2002.

[16] A.; Wan D. Bardram, J. E.; Mihailidis. *Pervasive Computing in Healthcare*. CRC Press, 2006.

[17] K-K. Lau, editor. *Component-Based Software Development: Case Studies*, volume 1 of *Series on Component-Based Software Development*. World Scientific Publishing Company, Cingapura, 2004.

[18] G. T. Heineman and W. T. Councill. *Component Based Software Engineering: Putting the Pieces Together*. Addison-Wesley, Boston, EUA, 2001.

[19] F. Bachmann, L. Bass, C. Buhman, S. Comella-Dorda, F. Long, J. Robert, R. Seacord, and K. Wallnau. Technical Concepts of Component-based Software Engineering. Technical Report CMU/SEI-2000-TR-008, Carnegie Mellon - Software Engeneering Institute, Maio 2000.

[20] G. Wang, L. Ungar, and D. Klawitter. Component Assembly for OO Distributed Systems. *Computer*, 32(7):71–78, Julho 1999.

[21] S Wagner. Zero-configuration of pervasive healthcare sensor networks. *Pervasive Computing and Applications, 2008. ICPCA 2008. Third International Conference on*, 1:405 – 408–2, Oct 2008.

[22] N.B Christensen and S Wagner. Using near field communication technology to achieve near-zero- configuration of sensors. Proceedings of MobiQuitous, 2008.