Arcabouço para o Desenvolvimento de Aplicações Pervasivas para Suporte à Prevenção e Tratamento de Doenças Crônicas

Mateus Assis Maximo de Lima

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

Arcabouço para o Desenvolvimento de Aplicações Pervasivas para Suporte à Prevenção e Tratamento de Doenças Crônicas

Mateus Assis Maximo de Lima

Dissertação de Mestrado apresentada em Fevereiro de 2010

Angelo Perkusich, D.Sc.
Orientador

Hyggo Oliveira de Almeida, D.Sc
Examinador

Antonio Marcus Nogueira Lima, Dr.
Examinador

Campina Grande, Paraíba, Brasil, Fevereiro de 2010

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL DA UFCG

L732a
2010

Lima, Mateus Assis Maximo.
Arcabouço para o desenvolvimento de aplicações pervasivas para suporte à prevenção e tratamento de doenças crônicas /Mateus Assis Maximo de Lima. — Campina Grande, 2010.
57 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)

ARCABOUÇO PARA O DESENVOLVIMENTO DE APLICAÇÕES PERVASIVAS PARA SUPORTE À PREVENÇÃO E TRATAMENTO DE DOENÇAS CRÔNICAS

MATEUS ASSIS MÁXIMO DE LIMA

Dissertação Aprovada em 19.03.2010

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

Dedicatória

Dedico esta dissertação a todos que fazem parte da minha vida. Principalmente a minha família: pai, mãe e irmãos; minha tia Zélia (em memória); minha cunhada; minha afilhada Luma e minha noiva Arissa.

Agradecimentos

Somos a soma das experiências de todas as pessoas que fazem parte da nossa vida.

Agradeço primeiramente a essa energia grandiosa que une a todos e nos faz ter força nos momentos mais difíceis. Agradeço, pai, a você: essa pessoa sem igual, dono de um conhecimento infindo da vida e das pessoas. Nossas conversas marcam minha vida. A você, mãe, por me apoiar, juntamente com toda nossa família, a todo momento. Muito obrigado por ser meu laço mais forte à crença no inexplicável, fazendo-me aceitar minhas limitações mas sempre procurar desvendar e entender o desconhecido. Aos meus irmãos, Rafael e Samuel, por serem sempre tão solícitos e amigos. A minha cunhada Luana, pela amizade e pelo presente maravilhoso que trouxe à minha família: Luma, minha afilhada. A Marcos Júnior, o melhor amigo que uma pessoa pode encontrar, que também faz parte dessa família.

A minha noiva, Arissa, pelo carinho, companheirismo e compreensão sem precedentes. Obrigado por suportar meus piores momentos e me fazer sempre descobrir o quanto sou humano e tenho no que melhorar.

Ao meu orientador, Angelo Perkusich, pelo seu imenso apoio, incentivo e amizade. Sua vontade de fazer as coisas acontecerem é inspiradora. A Hyggo que, apesar de não figurar como co-orientador, me instruiu a todo momento, indicando os caminhos a serem seguidos e corrigindo os meus erros. A Antonio Marcus pelo exemplo de pessoa e profissional e a Marcos Morais, grande amigo, de conversas longas e de um conhecimento irrestrito, que sempre semeia e nutre minha criatividade.

A minha tia Zélia (em memória), pelo exemplo de vida e coragem. Sua bela voz permeou toda minha vida e ainda ecoa em nossas almas. A toda minha família, principalmente aos meus tios Abel e Suely, a quem considero meus segundos pais.

A todos os meus amigos. Àquelas amizades, que mesmo à distância, não enfraquecem. Aos amigos do dia-a-dia e a aqueles que encontro ocasionalmente. Todos são muito importantes e contribuíram de alguma forma com este trabalho.

À Signove, por compreender a necessidade deste trabalho, apoiando-me irrestritamente, e a todos que a compõem.

Por fim, a todo aquele que já se sentou em uma mesa de bar comigo. Aos amantes da música e de um bom bate-papo. É sempre bom reencontrá-los.

Resumo

O atual paradigma de cuidado com a saúde já não suporta mais o crescente número de doentes crônicos. Uma vez que essas doenças apresentam um fator comportamental bem determinante, é necessário um monitoramento contínuo da saúde dos usuários. O monitoramento contínuo é inviável com o atual modelo de cuidados mundial. No tocante a isso, uma das abordagens mais promissoras é o autogerenciamento. Neste paradigma, o paciente se torna mais responsável pelo seu tratamento, tornando-se capaz de levar uma vida mais independente e desonerando o sistema de saúde.

Observa-se portanto a possibilidade de utilizar dispositivos que acompanhem o usuário a qualquer lugar e a todo momento no contexto da saúde. Essa idéia define uma área de conhecimento denominada Pervasive Healthcare. Diversas abordagens vem sendo desenvolvidas neste sentido. Várias delas são estudadas neste trabalho e observa-se que um dos grandes problemas é a falta de interoperabilidade entre componentes que definem estas aplicações. Soluções são desenvolvidas sem observar maneiras de maximizar a reutilização de módulos por diversas aplicações. Isto implica num custo maior no desenvolvimento e acarreta numa grande necessidade da elaboração de arcabouços de software que disponibilizem mecanismos para tal.

Neste trabalho apresenta-se um arcabouço para o desenvolvimento de aplicações pervasivas para suporte à prevenção e tratamento de doenças crônicas baseado em componentes de software. Ele visa dar suporte ao autogerenciamento utilizando conceitos de Pervasive Healthcare provendo uma ferramenta que facilite o desenvolvimento de aplicações e componentes de software que representam funcionalidades normalmente presentes em sistemas desse tipo. Além disso, leva-se em conta o suporte à evolução dinâmica da aplicação, além da implementação do arcabouço em linguagem multiplataforma para que possa ser executado em diferentes tipos de dispositivos móveis.

Por fim, para guiar o desenvolvedor na utilização do arcabouço para o desenvolvimento de aplicações e de componentes, descreve-se o processo de desenvolvimento através de um estudo de caso.

Abstract

The current paradigm of health care can no longer endure the growing number of chronically ill. Since these diseases have determinant a behavioral factor, a continuous monitoring of the health of users is necessary. Continuous monitoring is not feasible with the current model of worldwide care. With regard to this, one of the most promising approaches is the self-management. In this paradigm, the patient becomes more responsible for their treatment and become able to lead a more independent life, relieving the health system.

It is observed therefore the possibility of using devices to monitor the user at any place and time in the context of health. This idea defines an area of knowledge called Pervasive Healthcare. Several approaches have been developed in this direction. Several of them are studied in this work and it is observed that a major problem is the lack of interoperability between components that define these applications. Solutions are developed without following ways to maximize the reuse of modules for various applications. This implies a higher cost in development and brings a great need for development of software frameworks that provide mechanisms for this.

This paper presents a framework for the development of pervasive applications to support the prevention and treatment of chronic diseases based on software components. It aims to support self-management using concepts of Pervasive Healthcare providing a tool that makes the development of application and software components that represent features normally found in such systems easier. Moreover, it takes account of the dynamic evolution support for the application, and the design of a cross-platform language framework that can be run on different types of mobile devices.

Finally, to guide the developer in using the framework for the development of applications and components, the process of development through a case study is described.

Sumário

Lista de Símbolos e Abreviaturas

BAN	*Body Area Network*
COMPH	*Componentes para Pervasive Healthcare*
DBC	*Desenvolvimento Baseado em Componentes*
EEG	*Eletroencefalograma*
GCV	*Gerente de Ciclo de Vida*
IMC	*Índice de Massa Corpórea*
m-Health	*Mobile Healthcare*
MOC	*Meta-Object Compiler*
OMS	*Organização Mundial de Saúde*
PH	*Pervasive Healthcare*
SMS	*Short Message Service*
VDA	*Verificador de Descrição da Aplicação*

Lista de Figuras

Capítulo 1

Introdução

Atualmente, segundo a *Organização Mundial de Saúde* (OMS), a principal causa de morte da população mundial são as doenças crônicas. A OMS define-as como "doenças de longa duração e de progressão geralmente lenta" (ORGANIZATION, 2010). Devido à sua longa duração, existem muitas oportunidades de prevenção, requerendo portanto uma abordagem de tratamento contínua e sistemática.

Doenças crônicas incluem doenças de coração, derrame, câncer, doenças respiratórias crônicas e diabetes. Deterioração visual e cegueira, deterioração auditiva e surdez, doenças orais e desordens genéticas são outras condições crônicas que respondem por uma porção significativa da carga global de doenças (WHO, 2005b).

A proliferação dessas doenças já atingiu patamares críticos. Em 2005, elas já respondiam por 35 milhões de um total de 58 milhões de óbitos, sendo o dobro do número de mortes causadas por todas as doenças infecciosas (incluindo Aids, tuberculose e malária), deficiências nutricionais combinadas e condições maternas. No Brasil previu-se que, em 2005, 72% das mortes seriam resultantes destas doenças, aproximadamente um milhão de pessoas. A distribuição das causas de morte no Brasil encontra-se ilustrada na Figura 1.1.

A OMS estima que se nenhuma ação contrária a essa ploriferação for tomada, de 2005 a 2015, 388 milhões de pessoas morrerão devido a estas doenças, sendo muitas dessas mortes prematuras (WHO, 2005b). O impacto macroeconônimo dessas mortes será substancial. A OMS estima que só em 2005 o Brasil perdeu 3 bilhões do PIB devido a mortes prematuras ocasionadas por problemas cardíacos, derrames e diabetes. Prevê-se ainda um aumento nessas perdas, ocasionando uma perda acumulada de 49 bilhões de dólares no período compreendido desde 2005 até 2015 (ABEGUNDE; STANCIOLE, 2006).

Através de vários estudos conduzidos em várias regiões do mundo, a OMS obteve provas sobre os fatores de risco das doenças crônicas. Esses fatores são conhecidos e um pequeno conjunto deles é responsável pela maioria das doenças crônicas tanto em homens como em mulheres em todas as regiões. Os fatores de risco modificáveis mais importantes

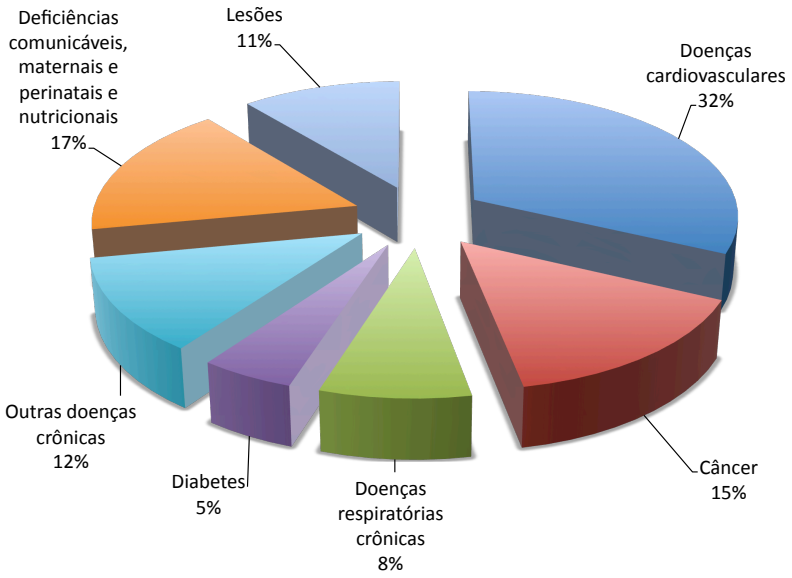

Figura 1.1: Distribuição das causas de morte projetadas para o Brasil em 2005 (WHO, 2009).

são uma dieta pouco saudável, consumo excessivo de energia, inatividade física e o uso do tabaco (Figura 1.2).

Esses fatores são expressos através dos fatores de risco intermediários como aumento da pressão sangüínea e do nível de glicose, número anormal de lipídios (particularmente o colesterol LDL), sobrepeso (*índice de massa corpórea* (IMC) , maior que 25 kg/m^2) e obesidade (IMC maior que 30 kg/m^2). Em conjunto com os fatores de risco não modificáveis, idade e hereditariedade, eles explicam a maioria dos novos casos de doenças cardíacas, derrames, doenças respiratórias crônicas e alguns tipos importantes de câncer.

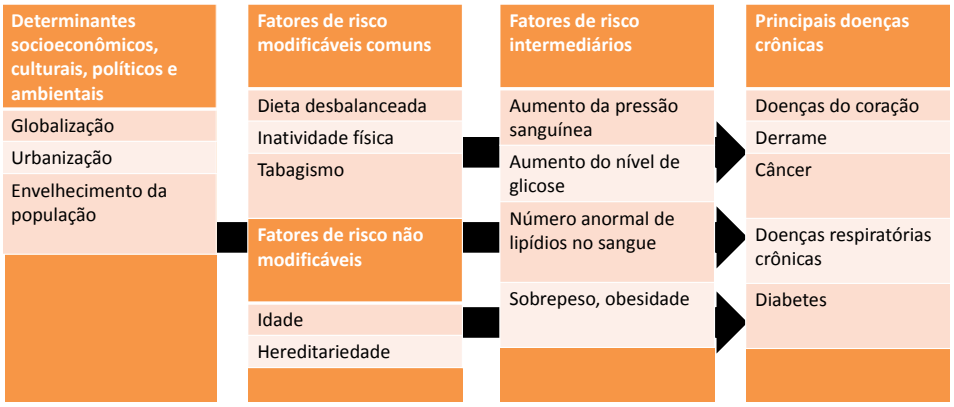

Figura 1.2: Principais fatores de risco de doenças crônicas (WHO, 2005a)

Pelo menos 80% das doenças prematuras do coração, derrames e diabetes tipo 2, e 40% das doenças de câncer podem ser prevenidas através de uma dieta saudável, atividade física regular e não utilização de produtos provenientes de tabaco (WHO, 2009).

Observa-se que, tanto para prevenir quanto para tratar doenças crônicas, requer-se um acompanhamento contínuo. Contudo, o custo de um acompanhamento deste tipo é muito alto, tendo em vista que necessita do envolvimento de: um profissional de enfermagem sempre com o paciente, para observar e fazer com que o tratamento seja seguido; um preparador físico, para a definição e acompanhamento das atividades físicas; um nutricionista para acompanhamento e definição da dieta; e um médico para gerenciar o tratamento.

Uma alternativa à utilização de tantos profissionais para o acompanhamento de um paciente é o autogerenciamento. Dessa forma, o paciente se torna mais responsável pelo seu tratamento, tornando-se capaz de levar uma vida mais independente e desonerando o sistema de saúde, uma vez que reduz o número de vezes que o paciente deve visitar um profissional de saúde ou ser internado. Há evidências substanciais em mais de 400 estudos de autogerenciamento que os programas que proporcionam aconselhamento, educação, retroalimentação e outros auxílios aos pacientes que apresentam condições crônicas estão associados a melhores resultados (OMS, 2003).

O autogerencimento porém faz com que os profissionais de saúde tenham menos informação sobre o paciente. Sendo assim, há grande relevância na utilização de sistemas que acompanhem o usuário a todo momento e em todo lugar, auxiliando no autogerenciamento do seu tratamento e das suas atividades possibilitando ainda o compartilhamento de informações acerca do paciente com os profissionais de saúde. Sistemas de autogerenciamento devem capturar dados de sensores médicos e disponibilizá-los em tempo hábil, criando canais pessoais de comunicação entre os profissionais de saúde e os pacientes. Além disso, tais sistemas ainda podem inferir estados de alerta através da realização de diagnósticos simples, comunicando-os aos profissionais responsáveis, tornando mais rápido o atendimento.

Os requisitos inerentes a sistemas de autogerenciamento são compatíveis com a recente evolução de pesquisa e desenvolvimento na área de Computação Pervasiva (BARDRAM; MIHAILIDIS; WAN, 2006). De acordo com este novo paradigma, tem-se computação em qualquer lugar e a qualquer momento, de modo transparente e autônomo, com o objetivo de prover serviços e recursos de interesse dos usuários. A computação pervasiva se apresenta como uma solução promissora para a implantação de sistemas de autogerenciamento, principalmente se considerados os avanços nas tecnologias de comunicação sem fio e dispositivos móveis.

A aplicação desses conceitos com a finalidade de adicionar qualidade de vida, saúde e bem-estar aos usuários, integrando os cuidados de saúde ao nosso cotidiano independentemente de espaço e tempo (BARDRAM; MIHAILIDIS; WAN, 2006), define uma das áreas de aplicação mais importantes para tecnologias pervasivas (WEIPPL; HOLZINGER; TJOA, 2006). Essa área é chamada *Pervasive Healthcare* (PH). Sua relevância remete princi-

palmente ao fato de não existir outro domínio onde a importância da tomada de decisão correta, baseada na obtenção da informação certa na hora certa, seja mais crítica (SATYANARAYANAN, 2001; WEISER, 1991). A PH dá suporte a esse modelo de cuidados pervasivo e distribuído resultante da implantação de sistemas de autogerenciamento, uma vez que mitiga o desconhecimento do prestador de cuidados através da utilização das tecnologias de informação e comunicação.

É nesse contexto de utilização de conceitos de PH para auxiliar o autogerenciamento de doenças crônicas que se insere o presente trabalho. Mais especificamente, pretende-se focar na prevenção e no acompanhamento do tratamento das doenças de coração, derrame, câncer, doenças respiratórias crônicas e diabetes, uma vez que observa-se o importante fator comportamental que faz com que a população as adquira.

1.1 Problemática

O grande potencial da utilização de PH no auxílio à implementação de um modelo de cuidados com a saúde mais distribuído, centrado no paciente e na sua educação, tornando-o mais responsável por seu tratamento, já tem sua importância sendo avaliada em diversos setores. A comunidade científica já notou esse potencial e as diversas lacunas de conhecimento que devem ser preenchidas. Por isso, observa-se atualmente um aumento no número de artigos publicados além de novos congressos a cada ano. A indústria, por sua vez, já vislumbra esse potencial e gigantes como a Microsoft, com o sistema HealthVault (MICROSOFT, 2010), e a Google, com o Google Health (GOOGLE, 2010), já apresentam soluções neste sentido. Além disso, ainda observa-se a formação de alianças de grandes empresas para estudar o potencial dessa área e desenvolver padrões de comunicação e representação de dados como no caso da Continua Health Alliance (CHA, 2010).

O domínio de aplicações de PH encontra-se em contínua expansão e muitas soluções vêm sendo desenvolvidas. Contudo, a maioria das soluções atuais como o *LifeShirt System* (VivoMetrics, Ventura, CA, USA) (VIVOMETRICS, 2009) e o *Philips TeleHealth Solutions* (Philips Medical Systems, Andover, MA, USA) (PHILIPS, 2009) são fechadas, no sentido de que disponibilizam todas as suas funcionalidades como uma solução proprietária, dificultando a integração com outros sistemas. Elas normalmente fixam as entradas que são possíveis, uma vez que dão suporte a apenas alguns sensores, além das saídas, uma vez que estão normalmente atreladas a apenas um serviço. Ou seja, as soluções não são desenvolvidas de forma a maximizar a reutilização de módulos por diversas aplicações. Há, portanto, uma grande necessidade da elaboração de arcabouços de software que disponibilizem mecanismos que facilitem esse reaproveitamento, diminuindo custos de desenvolvimento.

Os arcabouços para desenvolvimento de software de maneira modular não assimilam as funcionalidades normalmente encontradas no contexto de PH, uma vez que são desenvolvidos com o intuito generalista, não levando em conta as peculiaridades de aplicações neste contexto. Além disso, aplicações de PH podem ser desenvolvidas abordando diversos aspectos de cuidados com a saúde como (BARDRAM; MIHAILIDIS; WAN, 2006):

- Bem-estar e boa forma (indivíduos saudáveis);
- Gestão e prevenção de riscos (indivíduos com um fator de risco para uma doença);
- Gestão da doença (pacientes com doença crônica);
- Gestão da doença aguda e alta precoce (pacientes com uma doença aguda);
- Auxílio à vida independente (idosos e deficientes).

Cada contexto citado tem seu próprio conjunto de funcionalidades. A importância relativa entre elas também difere de aplicação para aplicação. Por exemplo, sob o contexto de gestão de doença aguda, a relevância de uma medição e o envio da mesma em tempo hábil é muito mais alta em comparação com aplicação no contexto de bem-estar, uma vez que a vida do paciente não está em risco se uma medição não for enviada para o médico no momento necessário.

O cuidado com a saúde é na maioria das vezes individualizado, podendo inclusive ser alterado com o tempo, o que acarreta em uma demanda por adaptação contínua de software para cada usuário. Acrescenta-se o fato de que, no contexto de PH, a disponibilidade de sensores é variante no tempo, além da medição não poder ser interrompida, o que demanda arcabouços de desenvolvimento de software que possibilitem alterações na aplicação em tempo de execução sem suspensão das medições que vêm sendo realizadas.

Tem-se a necessidade de um arcabouço que leve em conta a heterogeneidade e dinamicidade das aplicações em PH, provendo uma fácil montagem das mesmas. As soluções apresentadas na literatura normalmente discutem apenas uma parte do sistema, tentando resolver problemas como a configuração de sensores, mas não se preocupando em prover mecanismos para a configuração, remoção e adição de novas funcionalidades. Além disso, os sistemas contemplam o envio dos dados capturados por sensores mas não o monitoramento contínuo do status do tratamento.

Diante do exposto, avalia-se que um dos maiores desafios associados ao desenvolvimento de sistemas pervasivos no contexto de saúde e bem-estar é a integração e a interoperabilidade de um diverso conjunto de componentes incluindo sensores, *middleware*, bancos de dados e serviços, bem como ferramentas para mineração e visualização dos dados (ELHELW et al., 2009).

1.2 Objetivo

O objetivo deste trabalho é a concepção de um arcabouço para o desenvolvimento de aplicações pervasivas para auxílio à prevenção e tratamento de doenças crônicas baseado em componentes de software, possibilitando a evolução dinâmica da aplicação. O arcabouço é desenvolvido para atender aos seguintes requisitos:

- Uma vez que aplicações para o cuidado com a saúde monitoram sinais vitais, elas não podem parar. De fato, alterações devem ser feitas em tempo de execução e devem acarretar em pouco ou nenhum efeito no funcionamento do sistema;
- A interoperabilidade deve ser possível para todas as funcionalidades apresentadas anteriormente. Todos os componentes podem ser trocados em tempo de execução e isso não deve afetar o comportamento da aplicação. Uma vez que a disponibilidade de sinais provenientes dos sensores é dinâmica, o arcabouço deve ser capaz de se reconfigurar assim que os sensores tornarem-se disponíveis seguindo as regras definidas pelo prestador de cuidados de saúde;
- Os componentes desenvolvidos devem ser específicos para PH, reduzindo assim o esforço necessário para reutilizá-los em várias aplicações.

No que diz respeito à plataforma e arquiteturas alvo, o arcabouço será multiplataforma e desenvolvido para que possa ser executado em dispositivos embarcados, onde as restrições de poder computacional são mais severas.

A validação do arcabouço será feita através de um estudo de caso desenvolvido utilizando o arcabouço multiplataforma Qt, uma vez que ele é o arcabouço que melhor combina um conjunto vasto de plataformas-alvo e o desempenho das aplicações, podendo ser executado em dispositivos cujo sistema operacional seja Embedded Linux, Mac OS X, Windows, Linux/X11, Windows CE/Mobile, Symbian ou Maemo (QT, 2010). Neste trabalho, o dispositivo escolhido para a validação é o Nokia N900, dada a prévia familiaridade com o mesmo e o seu sistema operacional, Maemo, bem como o seu poder computacional e dispositivos embutidos nesta plataforma, o que o torna um dos dispositivos tecnologia mais avançada atualmente.

A aplicação desenvolvida para a validação deve, a partir da descrição em alto nível da aplicação elaborada para refletir as necessidades do prestador de cuidados, configurar todo o sistema de maneira transparente ao usuário assim que os sensores necessários estejam disponíveis. O prestador de cuidados deve estar sempre ciente do estado atual de monitoramento pois dessa forma, mesmo se o paciente não estiver seguindo o tratamento, ele estará informado a respeito. Assim, pode tomar medidas paliativas mais precocemente

que no modelo de saúde atual, onde os médicos só tomam conhecimento numa próxima consulta, o que em determinados casos pode ser tarde demais.

1.3 Relevância

O arcabouço proposto neste trabalho é um avanço no estado da arte uma vez que possibilita a adição, alteração e remoção de funcionalidades de forma contínua, reduzindo o impacto sobre a arquitetura e o código das aplicações existentes bem como o custo de desenvolvimento das mesmas. A falta de interoperabilidade, uma dos principais problemas existentes atualmente nas aplicações pervasivas no contexto de saúde e bem-estar é abordada, uma vez que todos os componentes são facilmente intercambiáveis sem a necessidade de parar ou reiniciar o sistema sempre que novos sensores estão disponíveis ou novos tratamentos são definidos, tornando o software adaptável continuamente a cada usuário.

No tocante à componentização das funcionalidades presentes na maioria das aplicações de PH, o arcabouço proposto abre a possibilidade da utilização de vários sensores presentes em diversas soluções fechadas. Para isso é necessário apenas o desenvolvimento de *drivers*, para comunicação com esses dispositivos, e a funcionalidade de envio das informações provenientes do sistema para um serviço, adicionando-os posteriormente como componentes do sistema. Desta forma, ao invés de ser necessária a compra de diversas soluções proprietárias apenas para a utilização dos serviços e sensores por elas providos, o usuário pode comprar um sensor de uma solução que disponibilize o sinal necessário e utilizá-lo em diversos serviços, reduzindo assim os custos para o usuário final.

A separação das funcionalidades como componentes de um sistema ainda possibilita o desenvolvimento de vários tipos de aplicações dentro dos diversos contextos de PH, uma vez que a montagem da aplicação é feita agregando componentes que representam funcionalidades da forma que o desenvolvedor desejar.

Além disso, dada a fácil intercambialidade entre componentes, o arcabouço proposto ainda possibilita o desenvolvimento de componentes separadamente e o teste dos mesmos em uma aplicação em execução. Isso abre caminho para o desenvolvimento de diversas áreas do conhecimento com um esforço de implementação reduzido focando apenas no problema a ser estudado, uma vez que as aplicações podem ser montadas utilizando estudos realizados em cada componente e a infra-estrutura necessária encontra-se disponível no arcabouço.

1.4 Estrutura da Dissertação

O restante deste documento está organizado da seguinte forma:

- No Capítulo 2 são apresentados conceitos pertinentes ao entendimento deste trabalho. Mais precisamente, são abordados conceitos relativos à Computação Pervasiva aplicada aos cuidados com a Saúde, Desenvolvimento baseado em Componentes e Arcabouço de Desenvolvimento Qt.
- No Capítulo 3 são apresentados trabalhos relacionados à infra-estrutura proposta.
- No Capítulo 4 descreve-se a infra-estrutura proposta, definindo componentes e seus contratos, bem como explicando o funcionamento do arcabouço em execução.
- No Capítulo 5 apresenta-se um estudo de caso da infra-estrutura apresentada no Capítulo 4.
- Capítulo 6 apresentam-se conclusões e perspectivas futuras deste trabalho.

Capítulo 2

Fundamentação teórica

Neste capítulo são apresentados conceitos considerados de alta relevância para o entendimento deste trabalho. Serão abordados aspectos relativos à Computação Pervasiva aplicada a área de cuidados com a saúde (*Pervasive Healthcare*), e ao Desenvolvimento Baseado em Componentes, enfatizando a definição de termos e conceitos que serão utilizados nos próximos capítulos.

2.1 *Pervasive Healthcare* e *Mobile Healthcare*

Nesta seção são apresentados conceitos relativos a *Pervasive Healthcare* e *Mobile Healthcare*. Inicialmente será realizada uma contextualização com relação à Computação Pervasiva, apresentando um breve histórico e seus conceitos fundamentais e, em seguida, situa-se o termo *Pervasive Healthcare* dentro desta área do conhecimento. São também abordados aspectos relacionados a *Mobile Healthcare*, as redes e formas de comunicação e as funcionalidades geralmente presentes em aplicações neste contexto. Por fim, descreve-se como os conceitos de *Pervasive Healthcare* são aplicados a este trabalho.

2.1.1 Computação Pervasiva

A computação pervasiva pode ser vista como a terceira geração dos paradigmas de computação. Na primeira, iniciada na década de 1960 e durando até os anos 1980, os computadores eram grandes *mainframes* compartilhados por vários usuários. Na segunda, o paradigma era centrado em computadores pessoais com usuários individuais. Na terceira, em ascensão desde o início do século XXI, cada usuário possui acesso contínuo à computação através de vários dispositivos em rede. A computação não mais se restringe a computadores pessoais de mesa. Na verdade, incluem-se dispositivos como telefones celulares, assistentes digitais pessoais (PDAs) e vários tipos de sensores. A interação com

esses dispositivos vai além da utilização de teclado, mouse e comandos de sistemas operacionais, sendo normalmente mais simples e as vezes até mesmo automática (BARDRAM; MIHAILIDIS; WAN, 2006).

Mark Weiser, considerado o "pai" da computação pervasiva, declarou que "as tecnologias mais profundas são aquelas que desaparecem. Tecem-se no tecido da vida cotidiana, até que são indistinguíveis dela." (WEISER, 1991). A computação pervasiva refere-se a conceitos da invisibilidade da computação, significando integração de poder computacional e sensoriamento em tudo, incluindo objetos do dia-a-dia (SATYANARAYANAN, 2001). Estes objetos se intercomunicam transparentemente, disponibilizando informações e recursos em qualquer lugar e a qualquer hora, de acordo com as necessidades e preferências dos usuários. O conceito chave da computação pervasiva, "em qualquer lugar e a qualquer hora", resume seus quatro princípios básicos: descentralização, diversificação, conectividade e simplicidade.

Descentralização - Em contraste com os primeiros paradigmas computacionais onde computadores executavam uma grande variedade de tarefas, a abordagem da computação pervasiva apresenta diversos dispositivos que executam tarefas mais especializadas. Serviços e aplicações são fornecidos por diversos dispositivos em rede que coordenam os seus recursos.

Diversificação - Uma vez que o paradigma da computação pervasiva emprega uma vasta gama de dispositivos, os quais podem interagir e compartilhar informações, os serviços são desenvolvidos para permitir que dispositivos futuros possam se comunicar com dispositivos atuais. Desta maneira, garante-se que as infra-estruturas criadas atualmente não limitem futuras possibilidades.

Conectividade - Interoperabilidade requer protocolos de comunicação que devem ser acordados por todas as partes que desenvolvem serviços e dispositivos para computação pervasiva.

Simplicidade - A computação pervasiva tem grande foco em aspectos relacionados à interação entre homem e máquina. Objetiva-se desenvolver sistemas que as pessoas possam aprender e usar facilmente, uma vez que estes sistemas devem tornar as formas de realizar tarefas mais práticas do que o modo habitual (BARDRAM; MIHAILIDIS; WAN, 2006).

2.1.2 *Pervasive Healthcare*

Uma das áreas de aplicação mais importantes para tecnologias pervasivas é a saúde, incluindo suporte à vida independente, bem-estar e gestão de doenças (WEIPPL; HOLZIN-

GER; TJOA, 2006). Cuidados com a saúde parece ser o terreno mais fértil para aplicações de computação pervasiva, haja vista que não há outro domínio onde a importância da tomada de decisão correta baseada na obtenção da informação certa na hora certa seja mais crítica (SATYANARAYANAN, 2001; WEISER, 1991). A aplicação das tecnologias de computação pervasiva nessa área é conhecida por *Pervasive Healthcare* (PH). Para a União Européia, este termo é equivalente a "Ambient Assisted Living" (SYMONDS, 2009). Tratam-se de cuidados de saúde disponíveis em todo o lugar e a qualquer hora. Em essência, *Pervasive Healthcare* abarca um conjunto de tecnologias relacionadas e conceitos que ajudam a integrar os cuidados de saúde mais perfeitamente ao nosso cotidiano, independentemente de espaço e tempo (BARDRAM; MIHAILIDIS; WAN, 2006).

Pervasive Healthcare (PH) refere-se às estruturas de interoperabilidade, invisibilidade, poder computacional e redes onipresentes que são empregadas com a finalidade de adicionar qualidade de vida, saúde e bem-estar aos usuários, estando eles saudáveis ou não. Ela envolve a interação individualizada entre serviços de saúde sobre uma camada de infra-estrutura de computação pervasiva (SYMONDS, 2009).

Em (WEISER, 1991), Weiser afirmou que para se alcançar o paradigma da computação pervasiva três elementos principais deveriam estar disponíveis: dispositivos baratos e com baixo consumo de energia; infra-estrutura de rede sem fio; e aplicações pervasivas. Na época em que esse artigo foi escrito, a tecnologia de hardware não estava preparada para dar suporte ao paradigma da computação pervasiva. O acesso às redes sem fio, ou não estavam disponíveis ou não eram possíveis através de dispositivos móveis. Consequentemente, aplicações pervasivas não eram desenvolvidas.

2.1.3 *Mobile Healthcare*

A partir dos avanços nas tecnologias embutidas em dispositivos móveis, a visão de Weiser tornou-se mais próxima da realidade. Atualmente, os dispositivos móveis possuem maior capacidade de memória e poder de processamento, o que os permite executar aplicações mais complexas, bem como diversas interfaces de comunicação com redes sem fio, tornando aplicações pervasivas mais viáveis.

A utilização de computação móvel, dispositivos médicos e tecnologias de comunicação — como celulares e computadores de mão — para serviços de saúde e informação é definida pelo termo *Mobile Healthcare* (m-Health) , introduzido pela primeira vez implicitamente como "Unwired e-med" na primeira edição especial do IEEE Transactions on Information Technology in Biomedicine (ISTEPANIAN; LAXMINARYAN, 2000).

Atualmente observam-se avanços significativos nas tecnologias de rede e comunicação sem fio em paralelo a avanços em sistemas pervasivos (JOVANOV et al., 2003; PATTICHIS et al., 2002). A evolução dos sensores e, geralmente, da tecnologia de medição, viabilizou a

obtenção de informações de saúde a partir de sensores embutidos, tanto dentro como fora dos hospitais, durante nosso dia-a-dia. A comunicação baseada em redes de telefonia móvel, redes locais sem fio e redes de sensores sem fio torna possível o acesso e transferência de todos os tipos de informação em qualquer lugar e a todo momento, incluindo informações relacionadas à saúde, tais como dados de medição ou de conhecimentos médicos (IEEE..., 2010).

2.1.4 BAN — *Body Area Network*

A rede de dispositivos formada pelo conjunto de sensores embutidos supracitados define o conceito de BAN (do inglês, *Body Area Network*). O conceito foi originalmente definido pela IBM (ZIMMERMAN, 1999) e desenvolvido posteriormente por outros pesquisadores. No livro *Book of Visions* do forum *Wireless World Research Forum*, BAN é definida como "uma coleção de dispositivos (inter)comunicáveis que estão junto ao corpo, proporcionando um conjunto integrado de serviços personalizados para o usuário" (FORUM; TAFAZOLLI, 2005).

A comunicação entre componentes de uma BAN é chamada comunicação intra-BAN. Se a BAN se comunica externamente, i.e. com outras redes (que também podem ser BANs), esta comunicação é vista, do ponto de vista da BAN, como uma comunicação extra-BAN(ISTEPANIAN; PATTICHIS, 2006).

Comunicação intra-BAN

A comunicação intra-BAN é realizada com tecnologias de média a curta distância, com ou sem fio. As opções com fio incluem fios de cobre e fibras ópticas além de muitas soluções de "computação vestível", onde circuitos são incorporados ou impressos em tecidos, ou incorporados a acessórios do dia-a-dia, tais como jóias ou óculos.

As soluções sem fio incluem infravermelho, micro-ondas, ondas de rádio e até mesmo a condutividade da pele. Dois padrões importantes para comunicação sem fio a curta distância são o *Zigbee* e o *Bluetooth*. *Bluetooth* é um exemplo de um padrão de comunicação sem fio (IEEE 802.15) que utiliza ondas de rádio e suporte velocidades até 721 kbps com alcance de até 100 m, além de oferecer uma forma simples de criar redes *ad hoc* possibilitando admissão e remoção de dispositivos em uma rede. *Zigbee* (ZigBee Alliance), também conhecido como RF-Lite, é um padrão de comunicação sem fio (IEEE 802.15.4) de baixo custo e baixo consumo de energia com alcance de até 50 m, que suporta uma taxa de dados relativamente baixa (até 250 Kbps), combinada com uma maior tempo de vida da bateria (ISTEPANIAN; PATTICHIS, 2006).

Apesar das características mais atraentes para a formação de BANs utilizando-se *Zig-*

bee, principalmente no tacante ao consumo de energia relativamente alto do padrão *Bluetooth*, este último encontra-se mais difundido atualmente, estando presente na maioria dos dispositivos móveis. Desta forma, uma grande parte das aplicações desenvolvidas no contexto de *m-Health* utilizam essa tecnologia.

Comunicação extra-BAN

Com o intuito de possibilitar conveniência e mobilidade para os usuários de redes BAN, a comunicação extra-BAN é geralmente baseada em tecnologia sem fio. Dentre as tecnologias disponíveis destacam-se Bluetooth, WLAN, GSM, GPRS e UMTS. A escolha da tecnologia de transmissão é feita dependendo de requisitos como distância, velocidade de conexão, segurança e confiabilidade.

BAN para Cuidados com Saúde

O conceito de BAN é genérico e pode ser aplicado a diversos domínios. Quando os dispositivos presentes em uma BAN medem sinais fisiológicos ou executam outras ações relacionadas ao cuidado com a saúde, define-se um tipo de especialização de BAN denominada h-BAN (do inglês, *health*-BAN).

No contexto de Cuidados com Saúde, a utilização de h-BANs combinadas com comunicações extra-BANs possibilita o monitoramento remoto de pacientes. Isso pode ser conseguido através de redes fixas, como telefonia fixa ou conexão com a Internet, permitindo ao paciente ser monitorado em casa ao invés de estar no hospital. Contudo, a utilização de redes fixas limita a mobilidade do paciente durante o monitoramento, ou no mínimo durante o envio dos dados obtidos armazenados localmente. Além disso, o armazenamento dos dados para um posterior envio só é viável quando não há requisitos de tempo real. Não obstante, esta escolha torna o sistema muito dependente da capacidade de armazenamento dos sensores, o que muitas vezes é bem restrita.

Por outro lado, combinando-se comunicações extra-BAN e h-BANs sem fio centradas em dispositivos móveis, abre-se possibilidade para diversas aplicações no contexto de PH, uma vez que os usuários possuem total mobilidade. Usuários podem manter suas atividades usuais, deixar suas casas e até mesmo viajar sem que o monitoramento seja interrompido (ISTEPANIAN; PATTICHIS, 2006).

2.1.5 Blocos Elementares para Aplicações em PH

Ao observar os sistemas desenvolvidos para o tratamento de saúde de forma pervasiva, nota-se que estes possuem tipicamente as seguintes funcionalidades (BONATO et al., 2006):

- *Sensores* para detectar sinais biomédicos ou parâmetros do usuário;

- *Métodos de comunicação a curta distância* para rotear os sinais provenientes dos sensores para um *gateway* que pode fazer algum processamento ou armazenamento e prover acesso ao usuário. Os *Sensores* em conjunto com os estes *Métodos* formam as h-BANs;
- *Algoritmos para processamento dos dados adquiridos pelos sensores*, com o intuito de extrair as grandezas médicas de interesse através, por exemplo, da remoção dos ruídos provenientes dos sensores;
- *Algoritmos de classificação e suporte à decisão* para diagnosticar uma condição e determinar a resposta adequada;
- *Métodos de comunicação a longa distância* para transmitir os dados para um serviço remoto. Tratam-se das comunicações extra-BAN citadas anteriormente;
- *Interfaces com os usuários* para interagir com todas as entidades relacionadas. Por exemplo, o paciente e um médico que está distante;
- *Sistemas de retaguarda* para possibilitar o armazenamento e o acesso aos consumidores dos dados;
- *Serviços* que provêem educação, aconselhamento e resposta ao profissional de saúde.

As aplicações que provêem essas funcionalidades, normalmente o fazem segundo um modelo "caixa preta" sem pontos de extensão, o que dificulta o reaproveitamento dessas separadamente em outras aplicações. Na próxima seção será apresentada uma técnica de desenvolvimento de software que leva em conta o encapsulamento de funcionalidades através de componentes de software. Essa técnica será utilizada mais adiante para o desenvolvimento do arcabouço proposto nesse trabalho.

2.1.6 PH e m-Health no Escopo do Trabalho

Neste trabalho, apresenta-se PH e m-Health como subáreas de conhecimento inseridas no contexto de Computação Pervasiva. Apesar disso, a evolução desses conhecimentos não se restringe apenas ao contexto da computação, envolvendo diversos outros profissionais como médicos, enfermeiros, engenheiros de hardware, redes de sensores, dentre outros.

PH e m-Health ainda possuem desafios não abordados neste trabalho que envolvem aspectos como segurança, privacidade, confiança, qualidade, etc. Apesar da grande relevância destes, foca-se na concepção de um arcabouço para desenvolvimento de aplicações no contexto de PH e um conjunto de mecanismos de engenharia que permitam a aplicação do mesmo.

Acredita-se que, com a base conceitual e ferramental proposta, o estudo desses aspectos em trabalhos futuros seja facilitado. Além disso, abre-se caminho para o desenvolvimento de cada funcionalidade separadamente sobre uma infra-estrutura em funcionamento. O desenvolvedor pode, por exemplo, focar-se no desenvolvimento de novos algoritmos para processamento dos dados adquiridos pelos sensores sem se preocupar em como a comunicação com os sensores será realizada, nem em como esses dados estarão disponíveis para os usuários através dos serviços.

2.2 Desenvolvimento Baseado em Componentes

Nesta seção abordam-se conceitos relacionados ao Desenvolvimento Baseado em Componentes (DBC) considerados necessários ao entendimento deste trabalho. São destacadas as principais entidades definidas neste paradigma, bem como aspectos relativos à flexibilidade, reutilização, configuração e disponibilização de componentes, e à interação entre eles. Por fim, descreve-se como esses conceitos são aplicados neste trabalho.

2.2.1 Principais Entidades

O Desenvolvimento Baseado em Componentes (LAU, 2004) é caracterizado pela composição de aplicações com base na reutilização de módulos de software pré-existentes (HEINEMAN; COUNCILL, 2001). Este processo de reutilização e montagem tem sido utilizado constantemente, promovendo um potencial aumento na produtividade do desenvolvimento e na qualidade da aplicação, diminuindo o tempo e o custo de produção (CRNKOVIC, 2001). Além disso, a utilização de DBC facilita o processo de evolução do software, uma vez que simplifica a inserção, alteração e remoção de funcionalidades.

O conceito central dentro do contexto de DBC é o de *componente*. Dentre as várias definições que podem ser encontradas na literatura (SAMETINGER, 2006; SZYPERSKY, 1998; NIERSTRASZ; TSICHRITZIS, 1995; JACOBSON, 1992; BOOCH, 1987; HEINEMAN; COUNCILL, 2001) utiliza-se, neste trabalho, a definição de Clemens Szyperski de que "um componente de software é uma unidade de composição com interfaces especificadas de forma contratual, podendo ser desenvolvida de forma independente e sujeita à composição por terceiros" (SZYPERSKY, 1998).

As abordagens de DBC geralmente são baseadas na arquitetura ilustrada na Figura 2.1. Nela observam-se dois outros conceitos utilizados na definição de componente: interface e contrato. A interface de um componente especifica quais são seus pontos de acesso, apresentando definições de métodos e valores que eles devem acordar, a fim de cooperar. Ela provê uma separação explícita entre definição e implementação, explicitando *o que* um componente implementa mas escondendo *como* ele o faz.

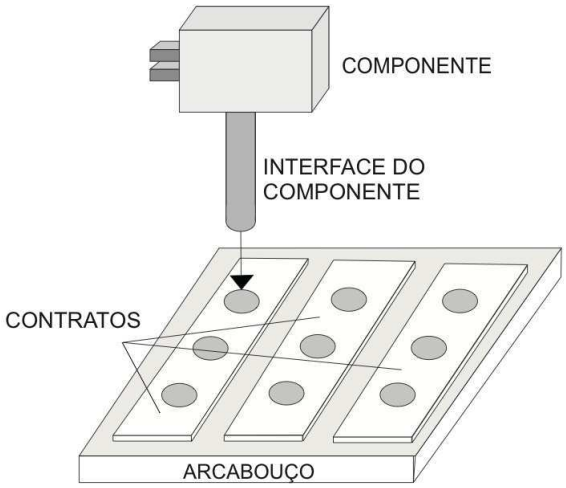

Figura 2.1: Arquitetura de Componentes (FERREIRA et al., 2004).

Um contrato representa a descrição de interface, ou seja, a especificação do comportamento de um componente (CRNKOVIC; LARSSON, 2002). 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 (BACHMANN et al., 2000). A especificação do comportamento de um componente, descrita no contrato, é constituído pelo invariante e a pré e pós-condições. O *invariante* são as restrições que o componente deve manter. As restrições que precisam ser satisfeitas pelo cliente para cada operação do componente são as *pré-condições*, enquanto as que o componente promete estabelecer são chamadas *pós-condições*.

Além da especificação do componente isoladamente, é necessária a definição de um conjunto de padrões e convenções para a construção de componentes e composição de aplicações baseadas em componentes. Esta especificação é denominada *modelo de componentes*. A infra-estrutura de suporte que impõe tais especificações, no nível de software, é denominada arcabouço de componentes (*component framework*) (BACHMANN et al., 2000). Os arcabouços vêm em geral acompanhados de uma infra-estrutura de gerência do ciclo de vida dos componentes chamada *middleware*.

2.2.2 DBC no Escopo deste Trabalho

Apesar dos diversos aspectos que podem ser estudados no contexto de DBC, como certificação e qualidade de componentes, o foco deste trabalho é a utilização de conceitos de DBC como interface, pré e pós-condições e componentes. Além disso, o modelo e arcabouço de componentes são definidos indicando as convenções utilizadas para a composição de aplicações e como essas são impostas no nível de software. Um middleware pervasivo é implementado para gerenciar o ciclo de vida dos componentes a partir da verificação de

sensores na proximidade, bem como da definição de novos tratamentos pelo prestador de cuidados.

Desta forma, o arcabouço possibilita a evolução dinâmica da aplicação levando-se em conta peculiaridades relativas ao contexto de PH, tornando-a adaptável à inserção, remoção ou alteração de equipamentos médicos, sensores e tratamentos. O arcabouço de componentes ainda verifica se a ligação entre os componentes pode ser realizada a partir da verificação do tipo de dado que cada componente envia e o que os outros podem receber.

2.3 Arcabouço Qt

Qt é um arcabouço multiplataforma para desenvolvimento de aplicações e interfaces de usuário. Ele possibilita o desenvolvimento de aplicações uma vez e sua execução em diversas plataformas, necessitando apenas a recompilação da aplicação para cada uma dessas plataformas.

O arcabouço Qt é, atualmente, o que melhor combina um conjunto vasto de plataformas-alvo e o desempenho das aplicações, podendo ser executado em dispositivos cujo sistema operacional seja Embedded Linux, Mac OS X, Windows, Linux/X11, Windows CE/Mobile, Symbian ou Maemo (QT, 2010).

Além de seu grande alcance com relação a plataformas-alvo, o arcabouço Qt provê mecanismos que facilitam o desenvolvimento baseado em componentos, tornando-o a melhor escolha para o desenvolvimento do arcabouço proposto neste trabalho.

2.3.1 Sinais e *Slots*

Durante o desenvolvimento de aplicações, muitas vezes necessita-se de mecanismos que possibilitam a comunicação entre objetos de diversos tipos. Por exemplo, quanto um usuário clica em um botão de Ok, provavelmente necessita-se que isto se reflita no comportamento da aplicação, obtendo, por exemplo, dados que foram preenchidos por esse usuário.

Este tipo de comportamento é normalmente implementado utilizando *callbacks*. Um objeto que necessite receber um evento de notificação se inscreve em um objeto como o seu observador, passando um ponteiro para outra função (*callback*) à função de processamento. Esta é responsável por chamá-lo na medida em que o evento ocorre. Contudo, essa abordagem possui dois problemas principais. Primeiro: não garantem que a função de processamento chame o *callback* com os argumentos corretos. Segundo: a função de *callback* é fortemente acoplada à função de processamento, uma vez que ela deve ter conhecimento do *callback* a ser chamado.

Com o intuito de mitigar esse problema, foi desenvolvido dentro do arcabouço Qt, uma alternativa: o mecanismo de sinais e *slots*.

Um sinal é emitido quando um evento ocorre e um *slot* é um função que é chamada em resposta a um sinal. Vários sinais podem ser conectados a um único *slot*, e um sinal pode ser conectado a vários *slots*. Juntos, sinais e *slots* formam um mecanismo muito poderoso de programação baseada em componentes.

Ao contrário do mecanismo de *callbacks*, o mecanismo de sinais e *slots* garante que a assinatura de um sinal deve combinar com a de um slot. Além disso, sinais e *slots* são fracamente acoplados, dado que uma classe que emite um sinal, não sabe nem se preocupa com o *slot* que receberá o sinal. Isso possibilita o encapsulamento da informação e garante que um objeto pode ser usado como um componente de software.

Este mecanismo é implementado utilizando um pré-compilador, chamado *MOC* (*Meta-Object Compiler*). O MOC analisa a declaração de classe, gerando um arquivo C++ e acrescenta código que inicializa o meta-objeto. O meta-objeto contém os nomes de todos os sinais e *slots*, bem como os ponteiros para estas funções. Além deste mecanismo, o MOC também implementa um sistema de propriedades dos objetos.

2.3.2 Propriedades

O arcabouço Qt oferece um sofisticado sistema de propriedades semelhantes às fornecidas por alguns outros compiladores. No entanto, como compilador e biblioteca são independentes da plataforma, Qt não pode depender de características não padronizadas de compiladores. A solução Qt funciona com qualquer compilador C++ padrão e em todas as plataformas suportadas por Qt.

As propriedades em Qt se comportam como dados-membros de classes, mas elas ainda possuem recursos adicionais, acessíveis através do sistema de meta-objetos. As propriedades têm nível de acesso definidos na sua declaração e podem ser lidas e escritas usando funções genéricas de um QObject, sem que seja necessário saber nada sobre a classe proprietária, exceto o nome da propriedade.

2.3.3 Qt no Escopo do Trabalho

No escopo deste trabalho são utilizados os mecanismos de sinais e *slots* na definição das entradas e saídas dos componentes. O sistema de propriedades é utilizado para definir configurações internas dos componentes, para que esses possam ser configurados através do arcabouço, utilizando funções genéricas, desacoplando assim o arcabouço das implementações dos componentes.

Capítulo 3

Trabalhos Relacionados

Diversas soluções vêm sendo propostas para superar a falta de interoperabilidade nos sistemas pervasivos para tratamento de saúde.Algumas destas soluções são descritas a seguir.

3.1 ElHelw

A arquitetura proposta em (ELHELW et al., 2009) baseia-se no modelo *push style message broker*. Neste trabalho, descreve-se uma plataforma para integração de múltiplos sensores de ambiente e sensores utilizados no corpo, algoritmos de fusão e análise de dados, numa arquitetura adequada à aplicações pervasivas no contexto de saúde. Ela é baseada num paradigma de troca de mensagens assíncronas, utilizando-se uma abordagem baseada em eventos. O processamento, a análise e fusão de dados para classificação dos perfis de comportamento dos usuários são efetuados em servidores dedicados.

3.2 Sarela

Em (SARELA; BIDARGADDI; KARUNANITHI, 2008), propõe-se uma arquitetura de sistema e um modelo de dados para executar o gerenciamento de informações essenciais para o acompanhamento ambulatorial em uma unidade de atendimento de saúde domiciliar. Essa arquitetura e modelo de dados são projetados para coletar dados ambulatoriais a partir de vários dispositivos e sistemas existentes, bem como para analisar, armazenar e apresentar informações clínicas significativas para o pessoal responsável pelo cuidado do usuário. A arquitetura proposta segue um modelo de desenvolvimento de sistema de informação em três níveis, descrito em (CHU; CESNIK, 2000). A interface do usuário é uma visão baseada em um navegador web, uma vez que o armazenamento de dados, análise e alertas estão disponíveis apenas em um servidor remoto. Não há informações a respeito

de um algoritmo de apoio à decisão, nem como o sistema é configurado pelo profissional de saúde. A estrutura de comunicação interna e o modelo de dados são baseados num repositório XML central que agrega os dados e possibilita a transferência de todos os dados entre os módulos principais e o banco de dados.

3.3 Guerra

Em (GUERRA, 2009), foca-se na definição e implementação da camada de conexão, descrevendo o desenvolvimento de uma prova de conceito de um sistema de estação base que detecta sensores na vizinhança via Bluetooth, carregando automaticamente todos os *drivers*. Além disso, o mecanismo descrito ainda descobre que sinais estão disponíveis a partir desses sensores e coloca-os em um banco de dados local. Uma visão geral deste trabalho pode ser vista na Figura 3.1.

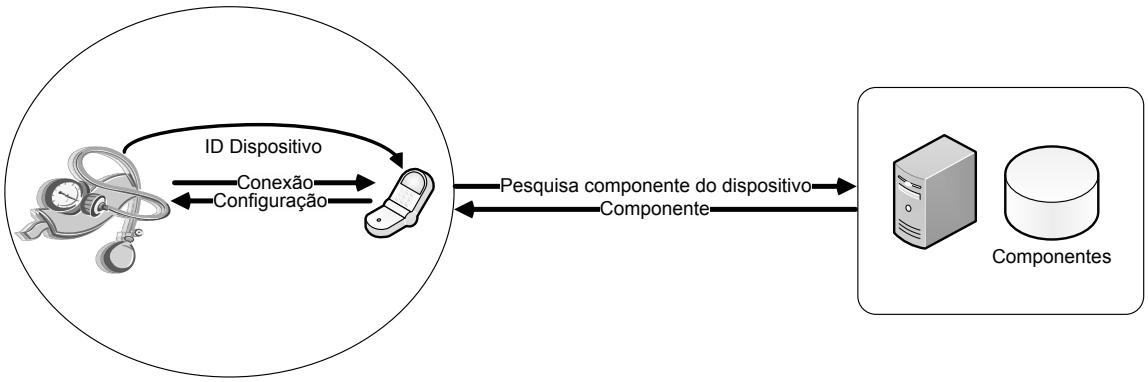

Figura 3.1: Mecanismo de Descoberta de Sensores

Este mecanismo é utilizado neste trabalho como ponto de partida para a montagem da estrutura da aplicação definida pelo desenvolvedor. Desta forma, na medida em que sensores estão disponíveis, a aplicação é montada e posta em funcionamento.

3.4 Wood

Em (WOOD et al., 2008), propõe-se um novo sistema de assistência ao dia-a-dia (do inglês, *assisted living system*) e de monitoramento residencial chamado AlarmNet. Ele integra sensores ambientais, fisiológicos e de atividade em uma arquitetura escalável e heterogênea. Um *middleware* de rede extensível e heterogêneo integra dispositivos embarcados, sistemas de retaguarda, análises remotas e interfaces de usuário. Ao passo que esse sistema só disponibiliza análise remota, o arcabouço proposto nesse trabalho provê uma forma de embutir o conhecimento de um profissional de saúde no sistema. Dessa forma, algumas análises e inferências de doenças e sintomas podem ser feitas localmente. Além

disso, o sistema proposto em (WOOD et al., 2008) não fornece os meios para disponibilizar informações de saída do sistema para diversos serviços.

3.5 PCOM

Em (BECKER et al., 2004), define-se um arcabouço generalista que utiliza o desenvolvimento baseado em componentes de software aplicados à Computação Pervasiva. Por se tratar de um arcabouço generalista, não são definidos componentes relativos ao contexto de PH.

O arcabouço PCOM utiliza uma descrição dos contratos dos componentes em XML e a conexão entre componentes é feita através do casamento entre interfaces. Um componente especifica a necessidade de utilizar uma determinada interface e outro indica que a disponibiliza.

O PCOM se baseia em um modelo onde os componentes pertencem sempre a contêiners que são responsáveis por gerenciar suas dependências. Ele possui mecanismos de adaptação que podem ser ou a interrupção da execução da aplicação ou a re-seleção de um componente em tempo de execução. A re-seleção em tempo de execução foi um requisito primário no desenvolvimento deste trabalho. Contudo, este procedimento é realizado verificando-se se os componentes atendem à especificação da aplicação.

Por se tratar de um arcabouço generalista, o PCOM não leva em conta as diferentes velocidades de execução de atividades por cada componente e nem demonstra a possibilidade de um armazenamento temporário dos dados na comunicação entre componentes.

Capítulo 4

Arcabouço COMPH

Neste capítulo detalha-se o arcabouço baseado em componentes para desenvolvimento de aplicações no contexto de Pervasive Healthcare COMPH (Componentes para Pervasive Healthcare). Inicialmente apresenta-se uma visão geral elucidando motivações que levaram à sua definição. Em seguida são definidos os componentes que fazem parte da arquitetura do arcabouço e, por fim, os mecanismos disponibilizados pelo arcabouço para a instanciação, adição, remoção e ligação de componentes, detalhando o processo de adaptação da aplicação quando esses eventos ocorrem. Além disso, ferramentas que auxiliam na execução da aplicação são estudadas, bem como a forma com a qual estas foram implementadas visando a gerência de recursos do sistema.

4.1 Visão Geral

O arcabouço de componentes COMPH baseia-se em conceitos relacionados à Computação Pervasiva aplicada à Saúde (*Pervasive Healthcare*) e Desenvolvimento Baseado em Componentes (DBC), além de ferramentas de programação disponíveis no arcabouço Qt, apresentados no Capítulo 2. Neste sentido, ele provê ferramentas que facilitam o desenvolvimento de aplicações levando em conta as funcionalidades tipicamente encontradas em aplicações no contexto de PH. O modelo de componentes baseia-se nessas funcionalidades definindo componentes que as refletem dentro do arcabouço, simplificando o entendimento da sua utilização no desenvolvimento de aplicações.

Além da definição de diversos tipos de componentes, o arcabouço ainda define alguns componentes utilitários que funcionam como suporte à aplicação gerenciando uso de memória e eventos, servindo como armazenadores temporários para a interligação entre componentes, além de possibilitar o agendamento de eventos através de um componente temporizador.

O arcabouço leva em conta as restrições mais severas de recursos encontradas em dispo-

sitivos móveis, buscando sempre minimizar sua utilização. O arcabouço provê mecanismos para que na medida em que recursos se tornem desnecessários, eles sejam liberados.

O COMPH ainda possibilita a elaboração de aplicações com diversas arquiteturas uma vez que não restringe a estrutura geral da aplicação, limitando-se a elaboração de mecanismos de ligação de componentes um a um, além de estruturas para a propagação de eventos, possibilitando a readaptação da aplicação quando da alteração dos componentes em atividade.

A implementação do COMPH utiliza o arcabouço de desenvolvimento de aplicações multiplataforma Qt, descrito na Seção 2.3. As entradas e saídas, tanto de dados, como de eventos, definidas para cada componente a seguir são implementadas utilizando o suporte a Slots e Sinais disponíveis em Qt.

4.2 Arquitetura

Diante das funcionalidades normalmente encontradas em aplicações de PH, define-se nesta seção alguns tipos de componentes, suas entradas e saídas, bem como suas permissões de acesso ao COMPH. Além disso, dada a heterogeneidade com que os dados são tratados por cada componente, componentes utilitários como memórias temporárias são definidos. Além disso, ferramentas de suporte para possibilitar temporização, envio de dados por diversos métodos de comunicação a longa distância e suporte a eventos para gerência da aplicação são apresentados.

4.2.1 Componentes

Ao observar os sistemas desenvolvidos para o tratamento de saúde de forma pervasiva, nota-se que estes possuem tipicamente as seguintes funcionalidades (BONATO et al., 2006):

- *Sensores* para detectar sinais biomédicos ou parâmetros do usuário;
- *Métodos de comunicação a curta distância* para rotear os sinais provenientes dos sensores para um *gateway* que pode fazer algum processamento ou armazenamento e prover acesso ao usuário;
- *Algoritmos para processamento dos dados adquiridos pelos sensores*, com o intuito de extrair as grandezas médicas de interesse através, por exemplo, da remoção dos ruídos provenientes dos sensores;
- *Algoritmos de classificação e suporte à decisão* para diagnosticar uma condição e determinar a resposta adequada;

- *Métodos de comunicação a longa distância* para transmitir os dados para um serviço remoto;
- *Interfaces com os usuários* para interagir com todas as entidades relacionadas. Por exemplo, o paciente e um médico que está distante;
- *Sistemas de retaguarda* para possibilitar o armazenamento e o acesso aos consumidores dos dados;
- *Serviços* que provêem educação, aconselhamento e resposta ao profissional de saúde.

Diante dessas funcionalidades apresentam-se a seguir os componentes que as implementam no contexto do arcabouço COMPH.

Comunicação a Curta Distância

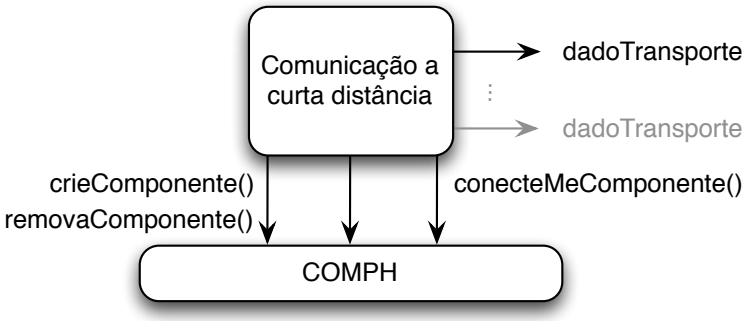

Figura 4.1: Componente de Comunicação a Curta Distância

O componente de Comunicação a Curta Distância é responsável por verificar a presença de sensores ao alcance do dispositivo e, a partir daí, identificar que *drivers* são necessários para iniciar a comunicação com esse tipo de dispositivo. A pesquisa é realizada inicialmente localmente, procurando por *drivers* que já foram utilizados anteriormente. Caso não exista um componente capaz de estabelecer essa comunicação, uma pesquisa em um servidor externo é realizada. Existindo um componente apto a promover esta conexão, ele é capturado do servidor e armazenado localmente, sendo depois instanciado.

Este componente tem acesso ao COMPH com permissão para requisitar a criação e remoção de componentes de Driver de Dispositivo, podendo também demandar uma posterior ligação com esse componente para que a configuração da comunicação possa ser realizada e a descoberta dos sinais disponíveis possibilitada.

Este componente encapsula métodos de comunicação que podem ser, por exemplo, Bluetooth, Infravermelho ou USB. Suas entradas são, portanto, essas interfaces de comunicação. Suas saídas são os dados capturados através destas interfaces.

A definição desta funcionalidade como um componente dentro do sistema possibilita que, mesmo que o acesso a estas interfaces seja normalmente dependente da plataforma, componentes possam ser desenvolvidos para cada dispositivo alvo, abstraindo a camada de comunicação com o sistema do desenvolvedor da aplicação, corroborando com o caráter multiplataforma para o qual o COMPH foi concebido.

Driver de Dispositivo

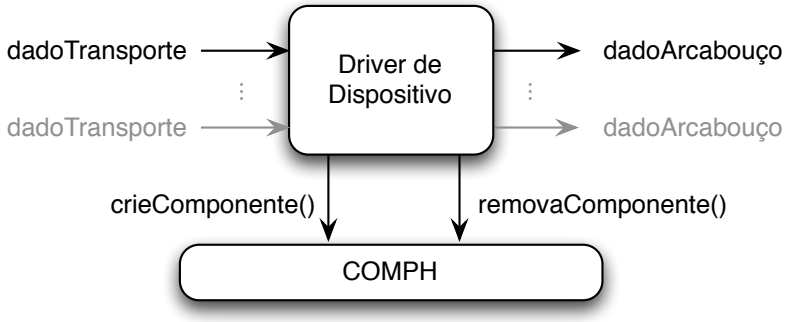

Figura 4.2: Componente Driver de Dispositivo

O componente de Driver de Dispositivo é responsável por se comunicar com os sensores através dos componentes de Comunicação a Curta Distância, estabelecer a comunicação desses com o sistema e descobrir os sinais que podem ser obtidos dos mesmos. Estes dados brutos ainda devem ser por ele interpretados de forma a terem significado dentro do contexto de PH. Desta forma, ele converte os dados de um formato de um fabricante ou de um determinado padrão para uma representação interna do arcabouço, referenciada internamente por um identificador único. Diante disso, outros componentes devem ter seu tipo de dado de entrada com esse mesmo identificador para que a conexão entre eles seja realizada com sucesso.

As entradas destes componentes são os dados disponíveis através das interfaces de comunicação, representadas no sistema pelos componentes de Comunicação a Curta Distância. Suas saídas são os dados já representados segundo o contexto de PH e definidos internamente através de um identificador único.

Este componente tem acesso ao COMPH com permissão para requisitar a criação e remoção de componentes do tipo Sensor. Essa requisição é resultante da descoberta com sucesso dos sinais disponíveis nos sensores, para que uma entidade de software os represente no arcabouço.

Essa funcionalidade de configuração e descoberta dos sinais disponíveis, realizada pelos componentes de Comunicação a Curta Distância em conjunto com o de Driver de Dispositivo é descrita em (GUERRA, 2009). Neste trabalho utiliza-se esse mecanismo como

ferramenta, focando-se na interação dele com outros componentes do arcabouço.

Um exemplo de funcionalidade que pode ser inserida num componente deste tipo é a implementação do padrão ISO/IEEE 11073 (IEEE, 2008). Esse padrão tem foco na conectividade *plug-and-play* de dispositivos biomédicos (BOTT et al., 2007). Ele ainda define especializações para diversos tipos de equipamentos como oxímetros, balanças, medidores de pressão, dentre outros. Desta forma, esses sensores poderiam ter suas representações no arcabouço, assim que a conexão fosse realizada com sucesso segundo este padrão.

A implementação desta funcionalidade como um serviço provido por um componente possibilita uma forma de comunicação com diversos dispositivos, cada qual com seu próprio protocolo de comunicação. Uma vez que o componente é carregado em tempo de execução, sua representação no sistema está disponível logo após a conexão estar estabelecida.

Sensor

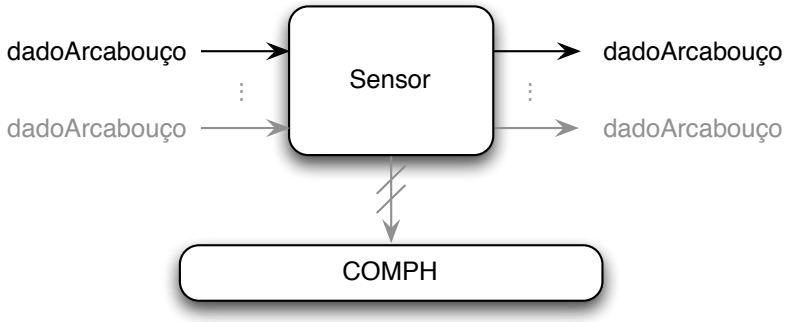

Figura 4.3: Componente Sensor

Uma vez que existem protocolos de comunicação que definem diversos tipos de componentes é necessária uma abstração que represente cada componente dentro do contexto do sistema. Estes componentes servem como interface entre o mecanismo descrito anteriormente e o arcabouço, uma vez que sua inserção no sistema é decorrente deste processo.

O componente Sensor é utilizado como peça chave para a execução do algoritmo de montagem da aplicação através do COMPH. A instanciação desse tipo de componente acarreta na avaliação da estrutura, outrora descrita pelo desenvolvedor da aplicação, para a montagem desta através da interligação dos componentes, como descrito no arquivo de configuração da aplicação.

Do ponto de vista de entrada e saída, estes componentes agem normalmente como repetidores, interligando cada entrada a uma saída com um mesmo tipo de dado. Além disso, não agem diretamente com o COMPH, uma vez que não necessitam instanciar nenhum outro componente.

Algoritmo para Processamento de Sinal

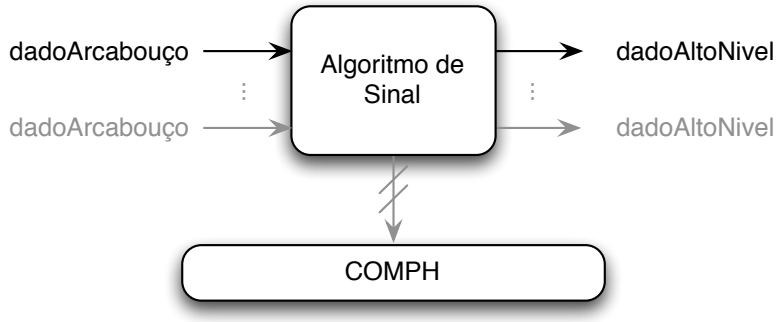

Figura 4.4: Componente Algoritmo para Processamento de Sinal

O tipo de componente Algoritmo para Processamento de Sinal recebe dados provenientes dos componentes de Sensor, e os processa, gerando em sua saída dados médicos de mais alto nível. A interligação com os componentes de Sensor só é realizada quando ele tem sua entrada definida com o mesmo identificador único que define o tipo de dado de saída do componente Sensor.

Um exemplo de funcionalidade que deve ser embutida em um componente desta natureza é o cálculo do complexo QRS a partir de um sinal de ECG. O dado proveniente do componente Sensor, em um cálculo deste, deve ser filtrado e alguma computação deve ser realizada para obter essa informação. Esse tipo de tratamento do sinal para a extração de informações mais pertinentes ao contexto de saúde é de responsabilidade deste tipo de componente.

Este componente pode ter várias entradas e saídas, podendo gerar, por exemplo, gráficos que auxiliem uma avaliação de um médico ou a classificação de algum estado através do componente Classificador.

Como resultado da implementação dessa funcionalidade como um componente, torna-se possível mudar em tempo de execução algoritmos deste tipo, bem como carregá-los apenas quando necessário. Os outros componentes não precisam ter conhecimento em como este dado está sendo extraído.

Classificador

O componente Classificador é responsável por receber dados, normalmente provenientes dos algoritmos de sinais ou até mesmo dos componentes Sensores, e inferir estados. Após a inferência, esse componente lança eventos que são normalmente capturados pelo componente de Algoritmo de Suporte à Decisão, descrito a seguir.

Apesar de estar normalmente agregado aos Algoritmos de Suporte à Decisão, optou-se por separá-lo no contexto deste trabalho para que esse último pudesse decidir a respeito

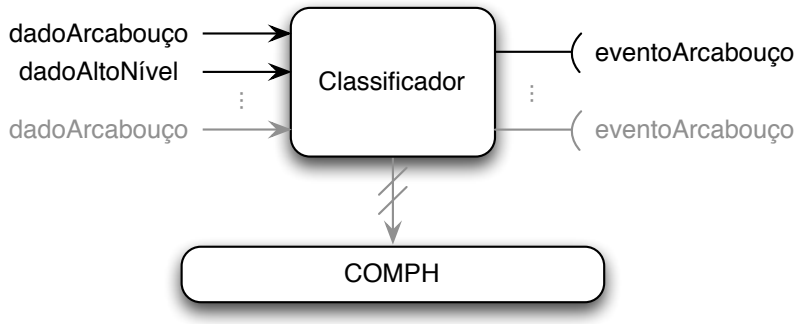

Figura 4.5: Componente Classificador

de aspectos não relacionados à classificação de estados médicos, bem como para garantir os diferentes níveis de acesso ao COMPH. Além disso, esta separação ainda propicia uma separação semântica, fazendo com que as regras de decisão permaneçam inalteradas, enquanto os algoritmos, que classificam os estados para que essas sejam disparadas, possam evoluir.

Do ponto de vista de entrada e saída, estes componentes recebem tipicamente dados e enviam eventos de mudança de estado. Além disso, não agem diretamente com o COMPH, uma vez que não necessitam instanciar nenhum outro componente.

Um exemplo de funcionalidade que pode ser embutida em um componente deste tipo é a classificação de que o usuário está potencialmente com gripe. A partir de medidas como a temperatura corporal e o nível de oxigenação do sangue, este estado pode ser inferido e um evento de mudança de estado propagado.

A implementação desta funcionalidade como um componente que pode ser alterado em tempo de execução, possibilita que o comportamento do sistema possa evoluir dinamicamente, na medida em que novas causas de doenças são descobertas ou novos métodos de classificação são desenvolvidos.

Algoritmo de Suporte à Decisão

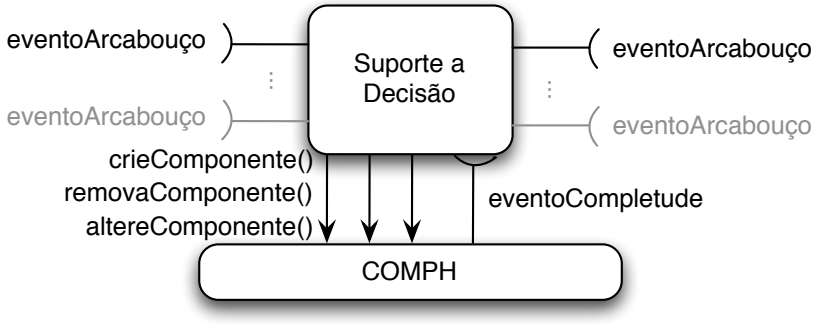

Figura 4.6: Componente Algoritmo de Suporte à Decisão

O componente do tipo Algoritmo de Suporte à Decisão recebe eventos de diversos componentes, como os componentes Classificadores ou o Gerente de Eventos, e, baseado em um conjunto de regras, dispara eventos como alarmes ou advertências para os componentes de Serviço. O retorno do estado de execução de uma regra é enviado através do componente Gerente de Eventos, fazendo com que este componente atualize o estado de disparo de uma regra e possa tomar outras decisões caso tenha tido sucesso ou não.

Além da possibilidade de comunicação com os componentes de Serviço, este componente possui permissão de acesso ao COMPH para requisitar a adição, alteração e remoção de componentes. Além dos componentes descritos nesta seção, este componente pode requisitar ao COMPH a alteração dos componentes de Conexão, descritos posteriormente. Desta forma, mudanças na configuração do sistema são propagadas e uma nova estrutura de aplicação é montada.

Este componente pode receber eventos de diversos componentes Classificadores e inferir se deve ou não enviar um alarme. Por exemplo, ele pode ser configurado para enviar uma alarme apenas se o usuário for classificado como gripado e com arritmia. Caso o usuário tenha sido classificado apenas como gripado, uma advertência pode ser enviada para outro serviço.

Uma vez conectado ao componente Gerente de Eventos, este componente pode disparar eventos de adicionar ou remover componentes caso tenha havido, por exempolo, um *buffer overflow* na conexão entre eles.

Uma vez que uma aplicação de PH recebe uma grande quantidade de dados provenientes de sensores, os prestadores de cuidado não têm tempo para interpretar tudo (LAMOTHE et al., 2006). É necessária uma filtragem para discernir quais informações são relevantes e quais precisam ser enviadas imediatamente ou ocasionalmente, por exemplo. Este tipo de configuração é inserida no sistema através deste tipo de componente. Como um componente, torna-se possível a melhoria do algoritmo para inferir quando o dado pode ser enviado, onerando menos os prestadores de cuidados. Além disso, a gerência de recursos do sistema pode ser realizada através da avaliação dos eventos gerados pelo sistema através do Gerente de Eventos, bem como pela definição de que componentes devem estar ativados a partir de regras inseridas pelo desenvolvedor da aplicação.

Serviço

O componente de Serviço é responsável por, a partir da recepção de um evento do Algoritmo de Decisão, empacotar os dados no formato do serviço para o qual foi desenvolvido e enviá-los através das interfaces de comunicação do dispositivo. Estas interfaces podem ser locais, como a própria tela do dispositivo, ou remotas, como interfaces de comunicação a longa distância.

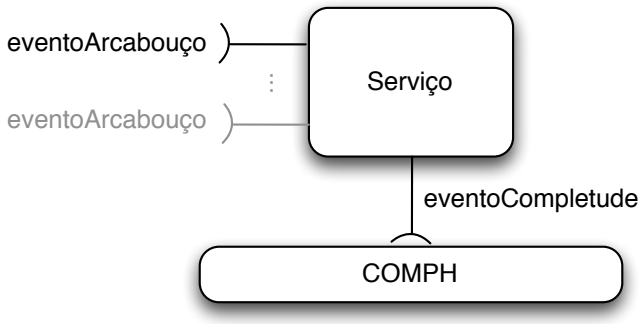

Figura 4.7: Componente de Serviço

Para enviar dados através dos meios de comunicação a longa distância presentes nos dispositivos, este componente normalmente faz uso do componente Gerente de Comunicação a Longa Distância, descrito a seguir, que recebe estes dados, e quando a interface estiver disponível os envia. Este último componente, por sua vez, atualiza o componente de decisão sobre o dado enviado para que este defina que o disparo da regra foi efetuado com êxito.

Este componente possui entradas de dados e de eventos. Os eventos definem como os dados devem ser enviados, e.g. como um alarme ou advertência. O componente, por sua vez, envia esses dados para o serviço para o qual foi desenvolvido. Além disso, estes componentes não interagem diretamente com o COMPH, uma vez que não necessitam instanciar nenhum outro componente.

Como exemplo, este componente pode enviar o peso do usuário para o serviço Vigilantes do Peso (WEIGHTWATCHERS, 2009). Ao mesmo tempo, outro componente deste tipo pode enviar esta mesma informação para o serviço TweetWhatYouEat (TWEETWHATYOUEAT, 2009). Além disso, um prestador de cuidados pode estar recebendo um alarme via SMS relativo a um estado de arritmia do usuário. Desta forma, é possível compartilhar dados provenientes de diversos sensores, bem como informações do sistema, com diversos serviços.

4.2.2 Componentes Utilitários

Com o intuito de dar maior suporte ao desenvolvedor das aplicações, o arcabouço COMPH fornece componentes utilitários. Estes componentes executam tarefas auxiliares visando, além de facilitar o desenvolvimento, a gerência mais eficiente dos recursos do sistema. A utilização destes componentes no sistema ou é resultado indireto da configuração das conexões entre dispositivos ou se dá através da execução de tarefas periféricas.

Gerente de Memória

O componente Gerente de Memória é responsável por garantir a integridade dos dados quando compartilhados por mais de um componente. As memórias temporárias utilizadas nas conexões podem ser compartilhadas desde que o Gerente de Memória garanta que os componentes que utilizam esses dados não os sobrescrevam, nem os descartem, sem que todos os seus consumidores já o tenham utilizado.

Esse componente ainda é responsável por armazenar os dados localmente quando ocorre *buffer overflow*, descartando-os posteriormente caso esses dados possuam um tempo de vida definido.

Gerente de Eventos

O componente Gerente de Eventos é acessível por todos os componentes do sistema e serve como interface para transmissão de eventos do sistema, bem como o registro destes para depuração. Ele pode ser conectado a componentes do tipo Serviço, possibilitando a depuração do sistema através, por exemplo, da Internet ou de um arquivo armazenado localmente.

Atualizador de Estado da Aplicação

Um dos principais objetivos do arcabouço é manter o prestador de cuidados sempre ciente do estado da aplicação. Sendo assim, cada vez que o arquivo de configuração é alterado no servidor, o prestador de cuidados se inscreve em um serviço que reporta o estado da configuração. Quando o arquivo é enviado para o arcabouço sendo executado no dispositivo do usuário através deste componente, ele é responsável por requisitar ao COMPH uma reavaliação da estrutura atual da aplicação, observando os sensores que estão disponíveis, podendo acarretar na alteração dos componentes em execução e suas ligações. Além disso, quando novos sensores estão disponíveis e o arcabouço tenta montar uma nova aplicação, este componente captura esses eventos e os envia para o servidor definido na configuração da aplicação. Dessa forma, o prestador de cuidados pode estar sempre ciente do estado da aplicação.

Temporizador

O componente Temporizador é responsável por receber inscrições de diversos componentes que desejam ser alertados quando um tempo determinado expirar. O componente que deseja receber eventos do temporizador deve disponibilizar uma entrada para receber o evento requisitado.

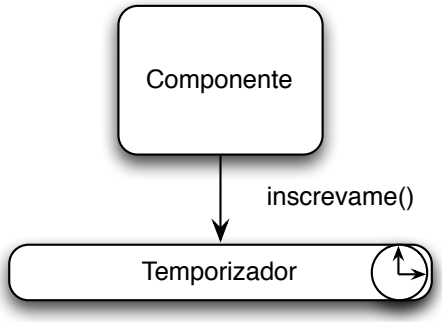

Figura 4.8: Componente Temporizador

Conexão

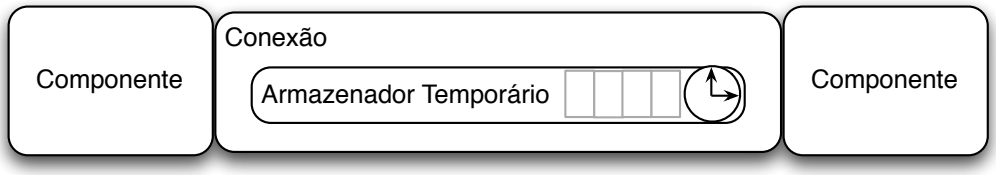

Figura 4.9: Componente de Conexão

O componente de Conexão representa a interligação entre dois componentes dentro do arcabouço. Ele possui um componente Armazenador Temporário para guardar dados e eventos a serem enviados de um componente para o outro. Ele pode ser ativado ou desativado pelo componente de suporte à decisão através do COMPH, sendo responsável por propagar esse evento para os componentes que ele interliga.

Armazenador Temporário (Buffers)

Uma vez que componentes podem executar atividades em diferentes velocidades é necessário um componente que armazene temporariamente os dados que possivelmente cheguem em uma velocidade maior do que o componente é capaz de processar. Trata-se de uma fila para armazenar dados e eventos que podem possuir um tempo de vida determinado. Caso este tempo se esgote, o componente Armazenador Temporário é responsável por descartá-lo ou requisitar ao componente Gerente de Memória, através do COMPH, o armazenamento local do dado. Ele aguarda pelo evento de mudança de estado do componente receptor do dado que indica que o componente já pode receber um novo dado, enviando-o posteriormente.

Gerente de Comunicação a Longa Distância

O componente Gerente de Comunicação a Longa distância recebe dados de diversos dispositivos através de um Componente de Conexão, procura por uma interface de longa distância disponível e os envia. Caso o envio não seja possível, envia um evento para o Gerente de Eventos indicando o erro para uma posterior depuração. Além disso, ele se inscreve no componente Temporizador para ser avisado periodicamente para iniciar um novo processo de busca até que o dado seja enviado.

4.2.3 Contratos

A especificação do comportamento de um componente, que garante 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, é representado dentro do arcabouço através de contratos.

O arcabouço de desenvolvimento Qt oferece ferramentas que facilitam a definição dessas estruturas. Para tal, utiliza-se principalmente o MOC (Meta-Object Compiler). As entradas e saídas dos componentes são definidas utilizando o suporte a Slots e Signals disponível em Qt, descritos na Seção 2.3. Propriedades adicionais como nível de qualidade de uma determinada saída ou requisito de nível de qualidade do sinal de entrada, são inseridos utilizando o suporte a Propriedades, também disponíveis em Qt.

4.3 COMPH em Execução

Nesta seção são abordados aspectos relativos aos componentes e ao arcabouço em tempo de execução. Inicialmente apresenta-se a máquina de estados que define o comportamento de cada componente. Logo após, define-se o arquivo de descrição da aplicação que pode ser alterado remotamente e como isto se reflete na execução. Em seguida, apresenta-se o middleware e como ele gerencia a execução da aplicação, definindo o comportamento da COMPH quanto à adição e remoção de componentes em tempo de execução.

4.3.1 Máquina de Estados Finitos dos Componentes

Com o intuito de especificar o modelo de comportamento dos componentes, o arcabouço COMPH define uma Máquina de Estados Finitos para cada componente. Na Figura 4.10, observam-se esses estados, bem como os eventos que disparam as transições entre eles. Cada estado implica em um comportamento bem específico, descrito a seguir:

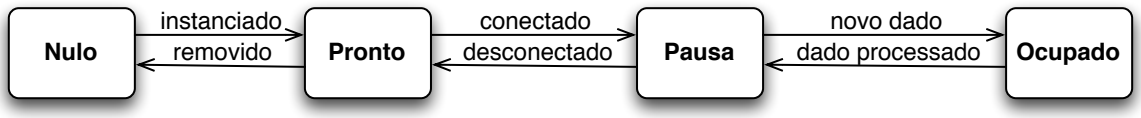

Figura 4.10: Máquina de Estados dos Componentes

Nulo: Estado inicial onde o componente ainda não foi instanciado. Neste estado, todos os recursos alocados para um componente estão desalocados.

Pronto: Estado onde o componente foi instanciado e configura os seus recursos internos, como interfaces de comunicação e memória. Assim, pode ter os tipos de dados de suas conexões verificados e já pode ser configurado através de suas propriedades. Neste estado, o fluxo de dados através das interfaces não está aberto ainda. Se esse fluxo tiver sido previamente aberto, ele deve ser fechado após ter recebido o evento de desconexão e transitado do estado de Pausa para o Pronto. Além disso, ao voltar para este estado, o componente tem suas propriedades e estado interno alterados para o seu valor inicial.

Pausa: Ao receber o evento de conectado o componente já pode receber dados. Isto ocorre porque antes desse evento ser enviado ao componente, o middleware já carregou os componentes de Conexão e os seus respectivos armazenadores temporários. Desta forma, toda a comunicação entre os componentes já pode ser realizada. Caso o componente utilize interfaces de comunicação, estas devem abrir seus fluxos de dados, possibilitando o envio de dados através delas.

Ocupado: Quando o componente recebe um evento de novo dado do componente de conexão, ele muda o seu estado para Ocupado quando vai processá-lo. Se o componente já estivesse previamente ocupado, um evento de componente ocupado é enviado para a conexão que pode, por sua vez, esperar pelo evento de mudança de estado do componente para o estado de Pausa para reenviá-lo. Além disso, o componente de conexão pode ainda se inscrever no componente Temporizador e requisitar para que após um tempo determinado seja feita uma nova tentativa. Sendo assim, este componente é responsável por, ao terminar o processamento de um dado, alterar o seu estado para o de Pausa, propagando esse evento para o componente de conexão.

4.3.2 Descrição da Aplicação

O desenvolvimento de aplicações utilizando o COMPH é realizado através da definição dos componentes e suas configurações, e a ligação entre eles. Essa escolha resulta em um

arquivo de descrição XML, como o que pode ser visto na Figura 4.11.

```
<COMPONENTES>
(a)     <COMPONENTE>
            <NOME = "ufcg.classificadorPressão"/>
(b)         <PROPRIEDADES>
                <QUALIDADE ENTRADA A>10</QUALIDADE ENTRADA A>
                ...
            </PROPRIEDADES>
            ...
        </COMPONENTE>
        ...
</COMPONENTES>
...
<CONEXOES>
(c)     <CONEXAO>
(d)         <ENTRADA>
                <COMPONENTE>ufcg.classificadorPressão</COMPONENTE>
                <METODO>novoDado(BYTE[])</METODO>
            </ENTRADA>
(e)         <SAIDA>
                <COMPONENTE>ufcg.medidorPressão</COMPONENTE>
                <METODO>dadoDisponivel(BYTE[])</METODO>
            </SAIDA>
(f)         <PROPRIEDADES>
                <MEMORIA>1000</MEMORIA>
                ...
            </PROPRIEDADES>
        </CONEXAO>
        ...
</CONEXOES>
```

Figura 4.11: Descrição da Aplicação

A partir da Figura 4.11, observa-se que a descrição da aplicação tem necessariamente duas seções principais: a seção que descreve os componentes (a) e a seção de conexões (c).

A seção de componentes (a) possui a definição de cada componente a ser utilizado no sistema, possuindo um campo obrigatório que é o seu nome e uma seção de propriedades, que é opcional. Nessa última descrevem-se propriedades internas dos componentes (b). Como exemplo, na Figura 4.11, descreve-se um componente classificador de pressão que indica que a qualidade necessária do sinal para a entrada A deve ser de valor 10.

Na seção das conexões (c), cada conexão entre os componentes definidos na seção de componentes (a) é definida. A conexão é definida explicitando-se que entrada de que componente (d) está ligada a que saída de outro (e). Além disso, podem ser definidas propriedades adicionais, como o tamanho da memória que deve ser alocada para essa conexão (f).

4.3.3 Middleware

O middleware do arcabouço COMPH é composto por dois componentes principais: o Gerente de Ciclo de Vida (GCV) e o Verificador de Descrição da Aplicação(VDA). Todas as requisições provenientes dos componentes passam inicialmente pelo verificador que avalia

as interconexões entre os componentes e as repassa para o gerente, caso seja necessária alguma intervenção no nível de componente.

O GCV é o responsável por, a partir da requisição de instanciação de um novo componente proveniente do COMPH procurá-lo localmente e, caso não o encontrando, capturá-lo de um servidor externo central e carregá-lo.

Através das propriedades definidas para cada componente, o GCV pode verificar que entradas são utilizadas por cada saída de um componente. Caso ele receba um evento do VDA indicando a remoção da conexão com qualquer entrada, ele é responsável por avaliar quais saídas são influenciadas, informando-o quais devem ser desativadas. Diante disso, outros componentes podem ser atingidos por essas modificações, uma vez que a saída desativada de um componente pode ser entrada desses outros.

Além da instanciação e verificação das alterações resultantes de uma reconfiguração da estrutura da aplicação, o GCV monitora continuamente se um componente ainda possui ligações ativas. Em caso negativo, ele é responsável por removê-lo automaticamente.

O VDA, por sua vez, é responsável pela verificação dos contratos dos componentes em tempo de execução, levando sempre em conta a definição da aplicação descrita pelo desenvolvedor para decidir se um componente deve ou não ser instanciado, removido ou alterado. Ele impõe a definição de um conjunto de padrões e convenções para a construção de componentes e composição de aplicações baseadas em componentes no nível de software. Além disso, o VDA ainda é responsável por definir o estado inicial de cada componente.

Uma vez que a configuração da aplicação pode ser alterada em tempo de execução, o VDA é responsável por, de posse da nova configuração, reavaliar todas as conexões, decidindo quais podem permanecer ativas e quais devem ser removidas. Após este procedimento, ele verifica novamente os componentes sensores que estão disponíveis e uma nova montagem da aplicação é realizada.

4.3.4 Funcionamento

Nesta seção será abordada a maneira pela qual o arcabouço reage à adição e remoção de componente em tempo de execução. São detalhados os algoritmos para propagação desses eventos em cada caso e como a aplicação se reconfigura.

Adição de Componentes

1. O arcabouço é iniciado e instancia o componente de Comunicação a Curta Distância;

2. Um sensor nas proximidades do dispositivo é detectado. O Componente de Comunicação a Curta Distância requisita ao COMPH a criação de um componente Driver

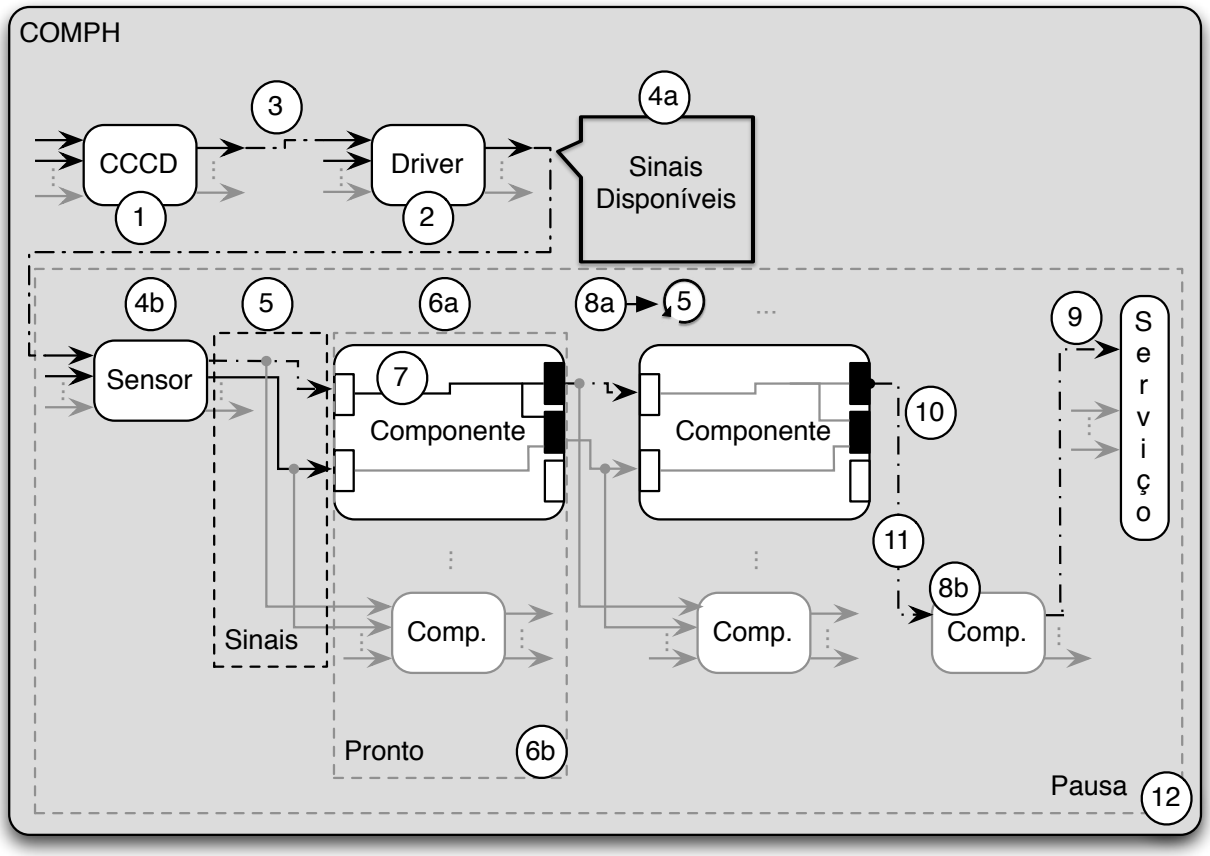

Figura 4.12: Funcionamento do Arcabouço - Adição de um Componente

de Dispositivo que possa lidar com este sensor.

3. O Componente de Comunicação a Curta Distância requisita ao COMPH a sua ligação a esse componente Driver de Dispositivo para que a comunicação com o sensor possa ser estabelecida.

4. Após a comunicação ser estabelecida com êxito, o componente Driver de Dispositivo descobre quais sinais podem ser extraídos deste sensor (4a). Ele requisita ao COMPH a criação de um componente Sensor que tem como saídas os sinais descobertos por ele (4b).

5. O Middleware verifica que entradas de que componentes estão ligados a cada uma das saídas dos componentes capturados no passo anterior através da seção CONEXOES do arquivo de descrição da aplicação.

6. Os componentes descobertos no passo 5 são capturados localmente ou remotamente (6a) e, posteriormente, instanciados, indo para o estado de Pronto (6b).

7. O COMPH verifica que saídas desses componentes são afetadas pela ativação das entradas descobertas no passo 5.

8. De posse dessas saídas, o Middleware volta ao passo 5 (8a) até que não existam mais componentes a terem suas saídas avaliadas (8b).

9. Uma vez que os componentes do tipo Serviço não possuem saídas a serem ativadas, esses componentes indicam o fim de um ramo da estrutura da aplicação. O COMPH avalia então que caminhos terminaram em componentes desse tipo.

10. Para cada caminho descoberto, o COMPH captura as configurações das conexões envolvidas e instancia os componentes de Conexão.

11. As conexões são ligadas aos respectivos componentes.

12. Os estados de todos os componentes são alterados para Pausa, iniciando dos componentes do tipo Serviço e terminando com os componentes Sensores.

Remoção de Componentes

Quando um componente é removido do sistema, essa alteração deve ser propagada na aplicação para que recursos desnecessários não continuem sendo utilizados. Para tal, o COMPH deve divulgar essa mudança desde o componente ou conexão até a entrada, que são os componentes de Comunicação a Curta Distância, e do componente até as saídas, que são os componente de Serviço.

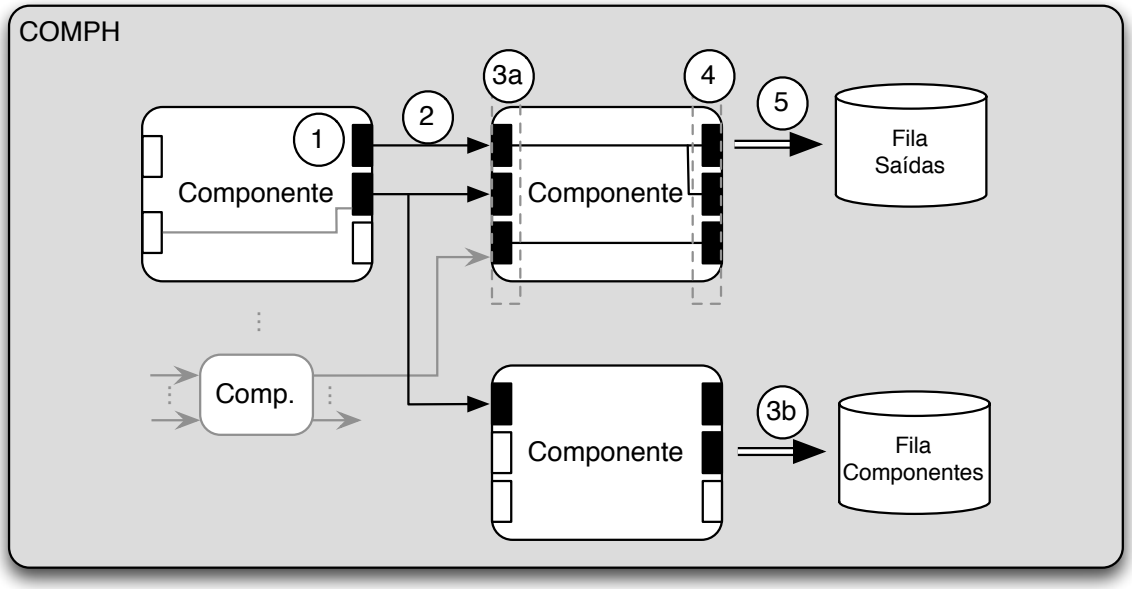

Figura 4.13: Propagação até as Saídas da remoção de um Componente

Propagação até as Saídas

1. O COMPH verifica que saídas estão ativas no componente que deve ser removido.
2. Logo após, ele verifica as conexões que envolvem essas saídas.
3. Para cada componente interligado a essa saída, verifica-se quais entradas estão ativas. Se a remoção da entrada envolvida na conexão descoberta no passo 2 acarretar em nenhuma entrada estar ativa, este componente é colocado numa fila a ser reprocessada, iniciando pelo passo 1.
4. Obtém-se uma lista de saídas ativas desse componente.
5. Verifica-se quais são as saídas ativas afetadas pela remoção da entrada descoberta no passo 3. Essas saídas são colocadas numa fila a ser reprocessada, iniciando pelo passo 2.
6. O algoritmo continua até não restar mais nenhum componente ou conexão a ser reavaliado nas filas descritas nos passos 3 e 5.

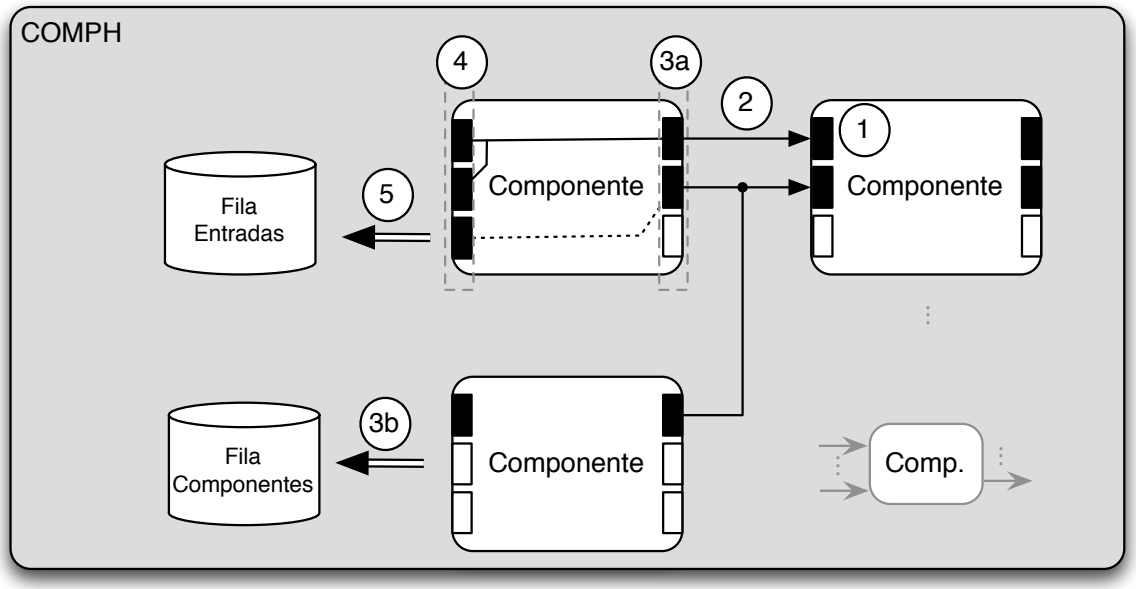

Figura 4.14: Propagação até as Entradas da remoção de um Componente

Propagação até as Entradas

1. O COMPH verifica que entradas estão ativas no componente que deve ser removido.
2. A seguir, ele verifica as conexões que envolvem essas entradas.
3. Para cada componente interligado a essa entrada, verifica-se quais saídas estão ativas. Se a remoção da saída envolvida na conexão descoberta no passo 2 acarretar em

nenhuma saída estar ativa, este componente é colocado numa fila a ser reprocessada, iniciando pelo passo 1.

4. Obtém-se uma lista de entradas ativas desse componente.

5. Verifica-se que entradas ativas se tornam desnecessárias pela remoção da saída descoberta no passo 3, haja vista que a computação por este componente realizada não terá mais saída pela qual ser enviada. Essas entradas são colocadas numa fila a ser reprocessada, iniciando pelo passo 2.

6. O algoritmo continua até não restar mais nenhum componente ou conexão a ser reavaliado nas filas descritas nos passos 3 e 5.

A utilização dos algoritmos descritos acima possibilita ao arcabouço uma readaptação, além de liberar recursos do sistema que não são mais necessários. Diante da premissa que o arcabouço pode ser utilizada em dispositivos móveis, este tipo de recurso é essencial, uma vez que estes dispositivos possuem restrições de recursos mais severas.

Capítulo 5

Estudo de Caso

5.1 Visão Geral

Para um melhor entendimento do arcabouço proposto e de como ele funciona é necessário definir os componentes e suas funcionalidade desempenhadas no sistema. Podem ser destacados os seguintes componentes-base:

- Sensores;
- Dispositivo Móvel;
- Servidor;
- Terminal do desenvolvedor;
- Terminal do prestador de cuidados.

Os sensores são responsáveis por coletar dados do usuário e enviá-los ao sistema através de um mecanismo de comunicação a curta distância, como por exemplo, o Bluetooth. O dispositivo móvel age como um terminal base, configurando a conexão com os sensores, processando os dados recebidos, inferindo estados médicos e, possivelmente, apresentando-os ao usuário, além de enviar estas informações através de métodos de comunicação a longa distância, como por exemplo, via Internet ou SMS (Short Message Service).

O servidor é responsável pelo banco de dados de definição das aplicações e de componentes. Esses devem ser obtidos, caso necessário, em tempo de execução pelo dispositivo móvel, que os utilizará para a montagem da aplicação. O terminal do desenvolvedor da aplicação é responsável por enviar a descrição da aplicação ao Servidor central que, ao recebê-lo, o envia para o dispositivo móvel. O dispositivo móvel é responsável também por manter atualizado o estado da aplicação no Servidor, que por sua vez pode compartilhar

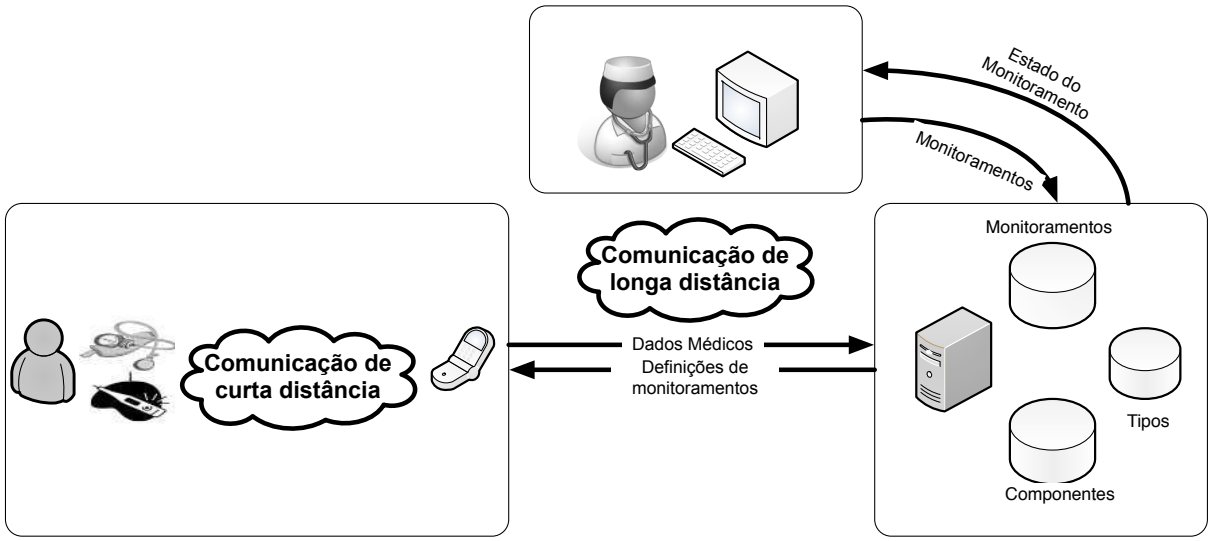

Figura 5.1: Visão Geral do Estudo de Caso

essa informação com o terminal do prestador de cuidados. Uma visão geral do sistema pode ser observada na Figura 5.1.

No contexto deste trabalho definiu-se a seguinte configuração para a validação do arcabouço (Figura 5.2):

Sensor: Foi utilizado o sensor ThinkGear-EM, TGEM2, da NeuroSky. Este sensor pode enviar dados brutos de EEG, bem como níveis de atenção e meditação (NEUROSKY, 2010).

Dispositivo Móvel: Foi utilizado o Nokia N900, dada a familiaridade com o desenvolvimento para esta plataforma, além deste já possuir suporte a Qt (NOKIA, 2010).

Servidor, Terminal do Desenvolvedor e do Prestador de Cuidados: Um computador comum com acesso a Internet.

A aplicação desenvolvida verifica o nível de sinais de EEG (Eletroencefalograma) e o reporta a um serviço Web continuamente. Caso o nível esteja abaixo de um patamar pré-determinado considerado aceitável, a aplicação envia um SMS ao médico. Além disso, existe um ramo da aplicação que captura o sinal bruto e o processa, enviando uma mensagem caso a média do dado bruto seja maior que um determinado valor. Essa aplicação tem grande utilidade numa área de conhecimento chamada BioFeedback (MONASTRA, 2005).

Estudos recentes (FRIEL, 2007; HAMADICHAREF et al., 2009; FOX; THARP; FOX, 2005) apresentam resultados promissores na aplicação desse conhecimento no tratamento do Transtorno por Déficit de Atenção com Hiperatividade. A aplicação dos conceitos de

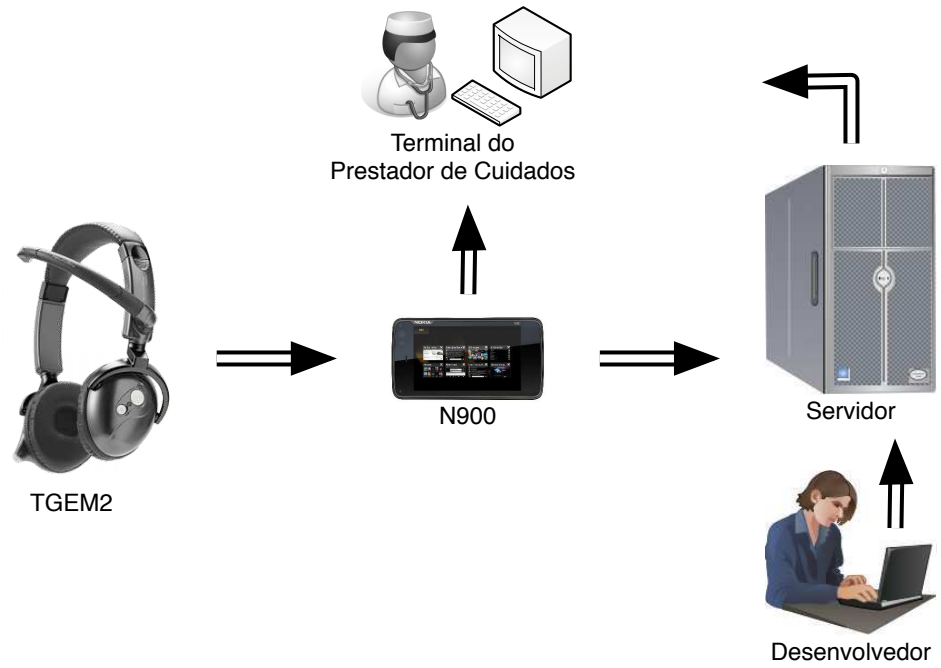

Figura 5.2: Visão Geral da Implementação

BioFeedback nesse tratamento utiliza o monitoramento do sinal EEG para indicar exercícios a serem executados no momento em que as amplitudes dos sinais de EEG alcancem valores que indicam um déficit de atenção.

De posse da definição do objetivo da aplicação e dos dispositivos a serem utilizados, a estrutura da aplicação, segundo os componente explicitados no Capítulo 4, é apresentada na Figura 5.3.

5.2 Desenvolvimento dos Componentes

Para exemplificar o processo de desenvolvimento de um componente escolheu-se, como prova de conceito, o componente driver de dispositivo para possibilitar o acesso aos dados provenientes do sensor de EEG.

Primeiramente devem ser identificadas as entradas e saídas do componente em questão. Neste caso, o driver do sensor de EEG deve receber dados através de uma interface de comunicação Bluetooth RFCOMM. Quanto às saídas, o driver poderá enviar os valores de EEG como dados brutos ou como dados de mais alto nível, como nível de atenção e de meditação. A escolha do tipo de saída será realizada através de uma propriedade deste driver.

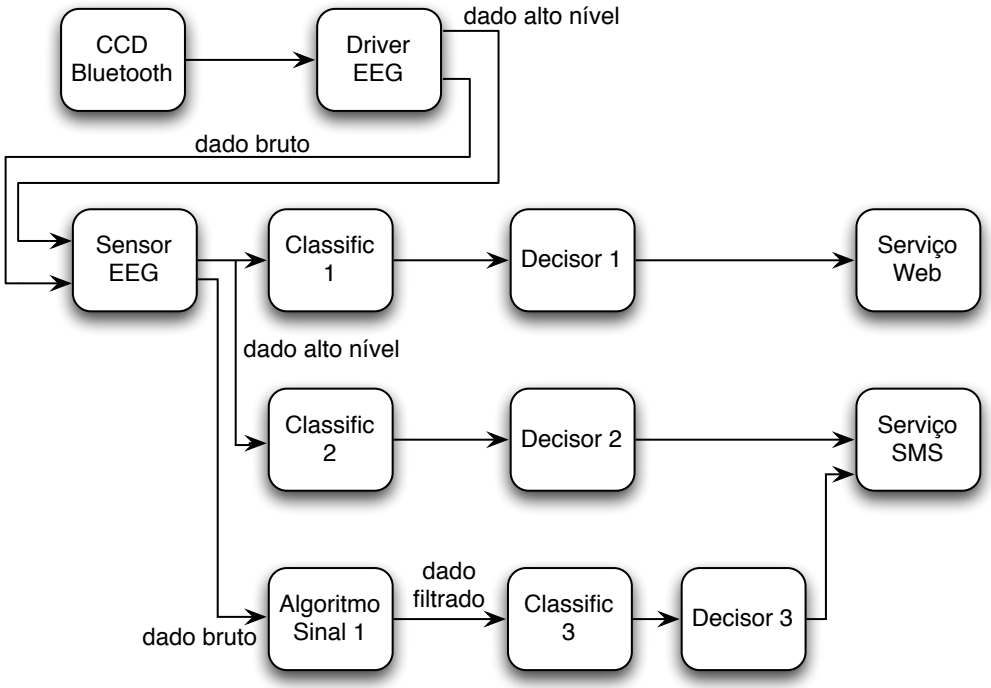

Figura 5.3: Componentes do Estudo de Caso

5.3 Desenvolvimento da Aplicação

Uma vez que os componentes estão disponíveis no servidor, o desenvolvedor da aplicação precisa apenas definir um arquivo de configuração como o apresentado na Figura 4.11. As conexões apresentadas na Figura 5.3 devem ser refletidas nesse arquivo. Além disso, configurações pertinentes a cada um do componentes devem ser descritas nos campos componentes dentro desses arquivos. O conteúdo do arquivo resultante desta especificação encontra-se descrito abaixo.

Código 5.1: Descrição XML da Aplicação

```
1  <COMPONENTES>
2    <COMPONENTE id="ufcg.sensorEEG.1">
3      <TIPO>ufcg.sensorEEG</TIPO>
4      </COMPONENTE>
5    <COMPONENTE id="ufcg.classificadorEEG.1">
6      <TIPO>ufcg.classificadorEEG</TIPO>
7      <PROPRIEDADES>
8        <TESTE>sempre</TESTE>
9      </PROPRIEDADES>
10     </COMPONENTE>
11   <COMPONENTE id="ufcg.classificadorEEG.2">
12     <TIPO>ufcg.classificadorEEG</TIPO>
```

```
13      <PROPRIEDADES>
14        <TESTE>"< 40"</TESTE>
15      </PROPRIEDADES>
16      </COMPONENTE>
17    <COMPONENTE id="ufcg.classificadorEEG.3">
18      <TIPO>ufcg.classificadorEEG</TIPO>
19      <PROPRIEDADES>
20        <TESTE>sempre</TESTE>
21      </PROPRIEDADES>
22      </COMPONENTE>
23    <COMPONENTE id="ufcg.decisorEEG.1">
24      <TIPO>ufcg.decisorEEG</TIPO>
25      <PROPRIEDADES>
26        <ENTRADA id="1">
27          <SAIDA>envieInfo</SAIDA>
28        <\ENTRADA>
29      </PROPRIEDADES>
30      </COMPONENTE>
31    <COMPONENTE id="ufcg.decisorEEG.2">
32      <TIPO>ufcg.decisorEEG</TIPO>
33      <PROPRIEDADES>
34        <ENTRADA id="1">
35          <SAIDA>envieAlarme</SAIDA>
36        <\ENTRADA>
37      </PROPRIEDADES>
38      </COMPONENTE>
39    <COMPONENTE id="ufcg.decisorEEG.3">
40      <TIPO>ufcg.decisorEEG</TIPO>
41      <PROPRIEDADES>
42        <ENTRADA id="1">
43          <SAIDA>envieInfo</SAIDA>
44        <\ENTRADA>
45      </PROPRIEDADES>
46      </COMPONENTE>
47    <COMPONENTE id="ufcg.algoritmoSinal.1">
48      <TIPO>ufcg.algoritmoMedia</TIPO>
49      <PROPRIEDADES>
50        <NUM_AMOSTRAS>10<\NUM_AMOSTRAS>
51      </PROPRIEDADES>
52      </COMPONENTE>
```

```
53    <COMPONENTE id="ufcg.servicoWeb.1">
54       <TIPO>ufcg.servicoWebComph</TIPO>
55       <PROPRIEDADES>
56          <SITE>http://www.comphmed.com.br<\SITE>
57       </PROPRIEDADES>
58       </COMPONENTE>
59    <COMPONENTE id="ufcg.servicoSMS.1">
60       <TIPO>ufcg.servicoSMS</TIPO>
61       <PROPRIEDADES>
62          <TEL>(83)555-1234<\TEL>
63       </PROPRIEDADES>
64       </COMPONENTE>
65 </COMPONENTES>
66
67 <CONEXOES>
68    <CONEXAO>
69       <ENTRADA>
70             <COMPONENTE>ufcg.classificadorEEG.1</COMPONENTE>
71          <METODO>entradaDadoAtencao(BYTE)</METODO>
72       </ENTRADA>
73       <SAIDA>
74          <COMPONENTE>ufcg.sensorEEG.1</COMPONENTE>
75          <METODO>novoDadoAtencao(BYTE)</METODO>
76       </SAIDA>
77       <PROPRIEDADES>
78          <MEMORIA>10</MEMORIA>
79       </PROPRIEDADES>
80    </CONEXAO>
81    <CONEXAO>
82       <ENTRADA>
83             <COMPONENTE>ufcg.classificadorEEG.2</COMPONENTE>
84          <METODO>entradaDadoAtencao(BYTE)</METODO>
85       </ENTRADA>
86       <SAIDA>
87          <COMPONENTE>ufcg.sensorEEG.1</COMPONENTE>
88          <METODO>novoDadoAtencao(BYTE)</METODO>
89       </SAIDA>
90       <PROPRIEDADES>
91          <MEMORIA>10</MEMORIA>
92       </PROPRIEDADES>
```

```
93  </CONEXAO>
94  <CONEXAO>
95      <ENTRADA>
96              <COMPONENTE>ufcg.algoritmoSinal.1</COMPONENTE>
97          <METODO>entradaDados(BYTE)</METODO>
98      </ENTRADA>
99      <SAIDA>
100         <COMPONENTE>ufcg.sensorEEG.1</COMPONENTE>
101         <METODO>novoDadoBruto(BYTE)</METODO>
102     </SAIDA>
103     <PROPRIEDADES>
104         <MEMORIA>100</MEMORIA>
105     </PROPRIEDADES>
106 </CONEXAO>
107 <CONEXAO>
108     <ENTRADA>
109             <COMPONENTE>ufcg.decisorEEG.1</COMPONENTE>
110         <METODO>entrada1()</METODO>
111     </ENTRADA>
112     <SAIDA>
113         <COMPONENTE>ufcg.classificadorEEG.1</COMPONENTE>
114         <METODO>regraDisparada()</METODO>
115     </SAIDA>
116     <PROPRIEDADES>
117         <MEMORIA>10</MEMORIA>
118     </PROPRIEDADES>
119 </CONEXAO>
120 <CONEXAO>
121     <ENTRADA>
122             <COMPONENTE>ufcg.decisorEEG.2</COMPONENTE>
123         <METODO>entrada1()</METODO>
124     </ENTRADA>
125     <SAIDA>
126         <COMPONENTE>ufcg.classificadorEEG.2</COMPONENTE>
127         <METODO>regraDisparada()</METODO>
128     </SAIDA>
129     <PROPRIEDADES>
130         <MEMORIA>10</MEMORIA>
131     </PROPRIEDADES>
132 </CONEXAO>
```

```
133  <CONEXAO>
134    <ENTRADA>
135          <COMPONENTE>ufcg.decisorEEG.3</COMPONENTE>
136        <METODO>entrada1()</METODO>
137    </ENTRADA>
138    <SAIDA>
139        <COMPONENTE>ufcg.classificadorEEG.3</COMPONENTE>
140        <METODO>regraDisparada()</METODO>
141    </SAIDA>
142    <PROPRIEDADES>
143        <MEMORIA>10</MEMORIA>
144    </PROPRIEDADES>
145  </CONEXAO>
146  <CONEXAO>
147    <ENTRADA>
148      <COMPONENTE>ufcg.classificadorEEG.3</COMPONENTE>
149        <METODO>entradaDadoMedia(BYTE)</METODO>
150    </ENTRADA>
151    <SAIDA>
152        <COMPONENTE>ufcg.algoritmoSinal.1</COMPONENTE>
153        <METODO>saidaMedia(BYTE)</METODO>
154    </SAIDA>
155    <PROPRIEDADES>
156        <MEMORIA>100</MEMORIA>
157    </PROPRIEDADES>
158  </CONEXAO>
159  <CONEXAO>
160    <ENTRADA>
161          <COMPONENTE>ufcg.servicoWeb.1</COMPONENTE>
162        <METODO>envieInfo()</METODO>
163    </ENTRADA>
164    <SAIDA>
165        <COMPONENTE>ufcg.decisorEEG.1</COMPONENTE>
166        <METODO>envieInfo()</METODO>
167    </SAIDA>
168    <PROPRIEDADES>
169        <MEMORIA>10</MEMORIA>
170    </PROPRIEDADES>
171  </CONEXAO>
172  <CONEXAO>
```

```
173         <ENTRADA>
174                 <COMPONENTE>ufcg.servicoSMS.1</COMPONENTE>
175             <METODO>envieAlarme()</METODO>
176         </ENTRADA>
177         <SAIDA>
178             <COMPONENTE>ufcg.decisorEEG.2</COMPONENTE>
179             <METODO>envieAlarme()</METODO>
180         </SAIDA>
181         <PROPRIEDADES>
182             <MEMORIA>10</MEMORIA>
183         </PROPRIEDADES>
184     </CONEXAO>
185     <CONEXAO>
186         <ENTRADA>
187                 <COMPONENTE>ufcg.servicoSMS.1</COMPONENTE>
188             <METODO>envieInfo()</METODO>
189         </ENTRADA>
190         <SAIDA>
191             <COMPONENTE>ufcg.decisorEEG.3</COMPONENTE>
192             <METODO>envieInfo()</METODO>
193         </SAIDA>
194         <PROPRIEDADES>
195             <MEMORIA>10</MEMORIA>
196         </PROPRIEDADES>
197     </CONEXAO>
198 </CONEXOES>
```

5.4 Aplicação em Funcionamento

Os dados provenientes do sensor de EEG são capturados pelo N900 que, através de um WebService, atualiza o conteúdo de uma página web. Esta página pode também ser acessada pelo prestador de cuidados. A interface desenvolvida está ilustrada na Figura 5.4.

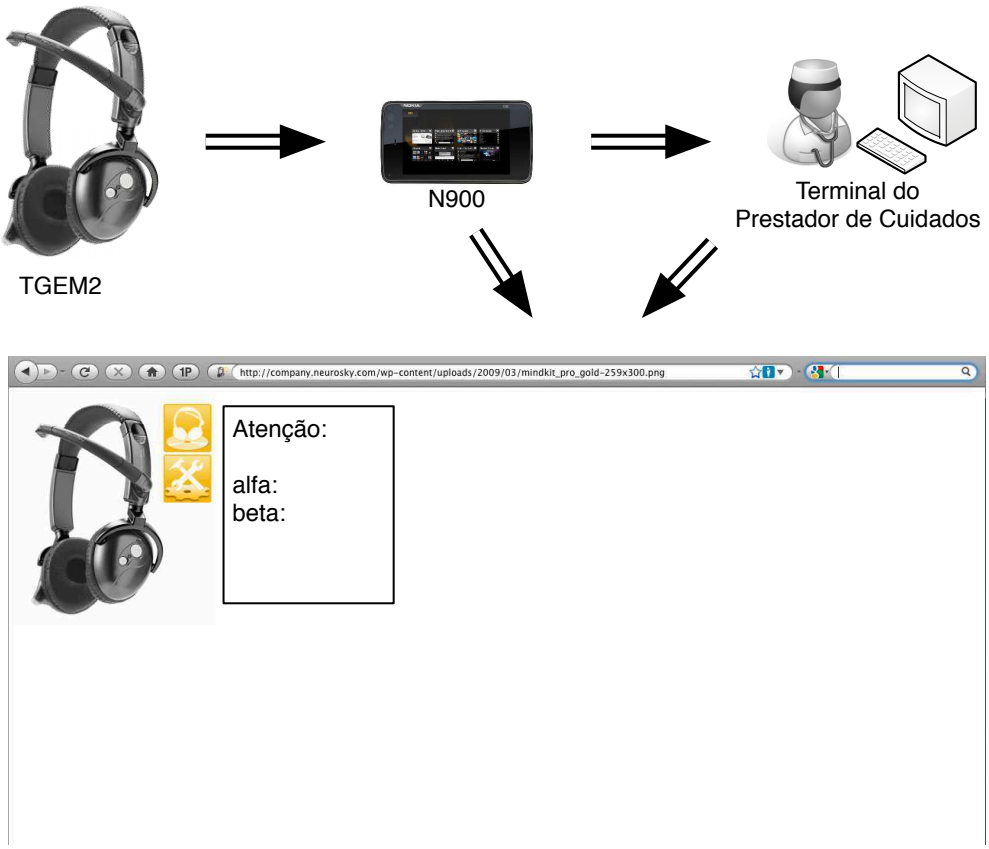

Figura 5.4: Funcionamento do Estudo de Caso

Capítulo 6

Conclusões e Trabalhos Futuros

O atual paradigma de cuidado com a saúde já não suporta mais o crescente número de doentes crônicos. Uma vez que essas doenças apresentam um fator comportamental bem determinante, é necessário um monitoramento contínuo da saúde dos usuários. Um acompanhamento prévio é necessário de forma a trabalhar com os fatores de risco e evitar que as pessoas as adquiram. Caso o usuário já apresente uma doença crônica, esse acompanhamento provavelmente durará o resto da vida, de forma a ensinar o paciente a viver com estas doenças com duração tão longa.

O monitoramento contínuo é inviável com o atual modelo de cuidados mundial. No tocante a isso, várias abordagens vêm sendo sugeridas. Uma das mais promissoras é o autogerenciamento. Neste paradigma, o paciente se torna mais responsável pelo seu tratamento, tornando-se capaz de levar uma vida mais independente e desonerando o sistema de saúde.

A idéia de utilizar dispositivos que acompanhem o usuário a qualquer lugar e a todo momento, aplicado ao contexto de saúde, define uma área de conhecimento denominada Pervasive Healthcare. Diversas abordagens vem sendo desenvolvidas neste sentido. Várias deles foram estudadas neste trabalho e observa-se que um dos grandes problemas é a falta de interoperabilidade entre componentes que definem estas aplicações.

As soluções atuais são desenvolvidas sem observar maneiras de maximizar a reutilização de módulos por diversas aplicações. Isto implica num custo maior no desenvolvimento e acarreta numa grande necessidade da elaboração de arcabouços de software que disponibilizem mecanismos para viabilizar o reúso.

O arcabouço desenvolvido neste trabalho visa dar suporte ao autogerenciamento utilizando conceitos de Pervasive Healthcare através da utilização de componentes, desenvolvido com o intuito de prover uma ferramenta que facilite o desenvolvimento de componentes que representam funcionalidades normalmente presentes em aplicações desse tipo. Ele foi desenvolvido para que fosse multiplataforma e suportasse uma evolução dinâmica

da aplicação, visto que aplicações neste contexto devem evitar ao máximo a interrupção, haja vista a natureza dos sinais que ela monitora.

Além de contribuir para o desenvolvimento de aplicações e estudos futuros utilizando o arcabouço definido, várias funcionalidades podem ser agregadas à infraestrutura proposta, bem como melhorias em componentes que não eram foco deste trabalho. Como exemplo, existem lacunas quanto à segurança e avaliação dos componentes a serem inseridos. Além disso, a elaboração de ferramentas que facilitem o desenvolvimento, gerando automaticamente o arquivo de configuração da aplicação, é um ponto que facilitaria ainda mais a adoção da solução proposta neste trabalho. Os mecanismos de atualização do funcionamento da aplicação através da mudança automática de componentes também podem ser avaliados.

Por fim, espera-se que, com o resultado do presente trabalho, novas aplicações e componentes funcionais possam ser desenvolvidos mais facilmente e que, com o crescimento do banco de dados de componentes, o arcabouço venha a ser adotado cada vez mais, evoluindo como ferramenta e auxiliando o desenvolvimento da área.

Referências Bibliográficas

ABEGUNDE, D.; STANCIOLE, A. *An estimation of the economic impact of chronic noncommunicable diseases in selected countries*. [S.l.], 2006.

BACHMANN, F. et al. *Technical Concepts of Component-based Software Engineering*. [S.l.], Maio 2000. v. 2, n. CMU/SEI-2000-TR-008.

BARDRAM, J. E.; MIHAILIDIS, A.; WAN, D. *Pervasive computing in healthcare*. [S.l.]: CRC Press, 2006.

BECKER, C. et al. Pcom a component system for pervasive computing. In: *PERCOM '04: Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PerCom'04)*. Washington, DC, USA: IEEE Computer Society, 2004. p. 67. ISBN 0-7695-2090-1.

BONATO, P. et al. Ieee embs technical committee on wearable biomedical sensors & systems: position paper. *Wearable and Implantable Body Sensor Networks, 2006. BSN 2006. International Workshop on*, p. 2, 2006.

BOOCH, G. *Software Component with ADA*. Redwood City, CA, EUA: Benjamin-Cummings Publishing Co., Inc., 1987.

BOTT, O. et al. Towards new scopes: Sensor-enhanced regional health information systems. *Methods of Information in Medicine*, Jan 2007.

CHA. *Continua Health Alliance*. 2010. Acessado em Fevereiro de 2010. Disponível em: <http://www.continuaalliance.org/>.

CHU, S.; CESNIK, B. A three-tier clinical information systems design model. *International journal of medical informatics*, Jan 2000.

CRNKOVIC, I. Component-based Software Engineering - New Challenges in Software Development. In: *Software Focus*. [S.l.]: Wiley, 2001. v. 4, p. 127–133.

CRNKOVIC, I.; LARSSON, M. (Ed.). *Building Reliable Component-Based Software Systems*. [S.l.]: Artech House Publishers, 2002.

ELHELW, M. et al. An integrated multi-sensing framework for pervasive healthcare monitoring. *Pervasive Computing Technologies for Healthcare, 2009. PervasiveHealth 2009. 3rd International Conference on*, p. 1–7, Apr 2009.

FERREIRA, G. V. et al. Uma abordagem baseada em componentes para a construção de edifícios virtuais. In: CONSCIÊNCIA, V. . (Ed.). *Proceedings of VII Symposium on Virtual Reality - SVR 2004*. São Paulo: [s.n.], 2004. v. 7, p. 279–290.

FORUM, W. W. R.; TAFAZOLLI, R. *Technologies for the Wireless Future: Wireless World Research Forum (WWRF)*. [S.l.]: Wiley, 2005. 580 p.

FOX, D. J.; THARP, D. F.; FOX, L. C. Neurofeedback: an alternative and efficacious treatment for attention deficit hyperactivity disorder. *Appl Psychophysiol Biofeedback*, v. 30, n. 4, p. 365–73, Dec 2005. Disponível em: <http://www.springerlink.com/content/y497605170267178/>.

FRIEL, P. N. Eeg biofeedback in the treatment of attention deficit hyperactivity disorder. *Altern Med Rev*, v. 12, n. 2, p. 146–51, Jun 2007.

GOOGLE. *Google Health*. 2010. Acessado em Fevereiro de 2010. Disponível em: <https://www.google.com/health/>.

GUERRA, W. Uma infraestrutura baseada em componentes para desenvolvimento de aplicações pervasivas para cuidados com a saúde. Proposta de Dissertação de Mestrado. 2009.

HAMADICHAREF, B. et al. Learning EEG-based Spectral-Spatial Patterns for Attention Level Measurement. In: *2009 IEEE International Symposium on Circuits and Systems (ISCAS2009)*. Taipei Taiwan: [s.n.], 2009. p. 1465–1468. Disponível em: <http://hal.inria.fr/inria-00441412/PDF/BHC-EEG-ISCAS.2009.pdf>.

HEINEMAN, G. T.; COUNCILL, W. T. *Component-Based Software Engineering: Putting the Pieces Together*. [S.l.]: Addison-Wesley, 2001.

IEEE. *Health informatics - Personal health device communication - Part 20601: Application profile - Optimized exchange protocol*. [S.l.], 2008.

IEEE 802.15 Working Group for WPAN. Jan 2010. Disponível em: <http://www.ieee802.org/15/>.

ISTEPANIAN, R. S. H.; LAXMINARYAN, S. Unwired, the next generation of wireless and internetable telemedicine systems-editorial paper. *Ieee T Inf Technol B*, v. 4, p. 189–194, 2000.

ISTEPANIAN, R. S. H.; PATTICHIS, C. S. *M-health: emerging mobile health systems*. [S.l.]: Springer, 2006.

JACOBSON, I. *Object-oriented Software Engineering*. New York, NY, EUA: ACM Press, 1992. ISBN 0-201-54435-0.

JOVANOV, E. et al. Stress monitoring using a distributed wireless intelligent sensor system. *Engineering in Medicine and Biology Magazine, IEEE*, v. 22, n. 3, p. 49 – 55, May 2003.

LAMOTHE, L. et al. Impacts of telehomecare on patients, providers, and organizations. *Telemedicine Journal & e-Health*, v. 12, n. 3, p. 363–369, 2006.

LAU, K.-K. (Ed.). *Component-Based Software Development: Case Studies*. Cingapura: World Scientific Publishing Company, 2004. (Series on Component-Based Software Development, v. 1).

MICROSOFT. *Microsoft HealthVault*. 2010. Acessado em Fevereiro de 2010. Disponível em: <http://www.healthvault.com/>.

MONASTRA, V. J. Electroencephalographic biofeedback (neurotherapy) as a treatment for attention deficit hyperactivity disorder: rationale and empirical foundation. *Child Adolesc Psychiatr Clin N Am*, v. 14, n. 1, p. 55–82, vi, Jan 2005. Disponível em: <http://www.childpsych.theclinics.com/article/S1056-4993(04)00067-7/abstract>.

NEUROSKY. *NeuroSky*. Fev 2010. Acessado em Fevereiro de 2010. Disponível em: <http://www.neurosky.com/>.

NIERSTRASZ, O.; TSICHRITZIS, D. (Ed.). *Object-oriented Software Composition*. Hertfordshire, Reino Unido: Prentice Hall International (UK) Ltd., 1995. ISBN 0-13-220674-9.

NOKIA. *Nokia N900*. Fev 2010. Acessado em Fevereiro de 2010. Disponível em: <http://maemo.nokia.com/n900/>.

OMS. *Cuidados Inovadores para Condições Crônicas: componentes estruturais de ação: relatório mundial*. [S.l.], 2003.

ORGANIZATION, W. H. *WHO | Cronic Diseases*. 2010. Acessado em Dezembro de 2009. Disponível em: <http://www.who.int/topics/chronic_diseases/en/>.

PATTICHIS, C. et al. Wireless telemedicine systems: an overview. *Antennas and Propagation Magazine, IEEE*, v. 44, n. 2, p. 143 – 153, Apr 2002.

PHILIPS. *Philips TeleHealth Solutions*. 2009. Acessado em Dezembro de 2009. Disponível em: <http://www.healthcare.philips.com/main/products/telehealth>.

QT, N. *Qt - A cross-platform application and UI framework*. Fev 2010. Acessado em Fevereiro de 2010. Disponível em: <http://qt.nokia.com>.

SAMETINGER, J. *Software Engineering with Reusable Components*. [S.l.]: Springer Verlag, 2006.

SARELA, A.; BIDARGADDI, N.; KARUNANITHI, M. A software architecture and data model for community-based healthcare environments. *Pervasive Computing Technologies for Healthcare, 2008. PervasiveHealth 2008. Second International Conference on*, p. 161–164, 2008.

SATYANARAYANAN, M. Pervasive computing: vision and challenges. *Personal Communications, IEEE*, v. 8, n. 4, p. 10 – 17, Aug 2001.

SYMONDS, J. *Ubiquitous and Pervasive Computing: Concepts, Methodologies, Tools, and Applications*. [S.l.]: Information Science Reference, 2009. 1819 p.

SZYPERSKY, C. *Component Software, Beyond Object-Oriented Programming*. [S.l.]: Addison-Wesley, 1998.

TWEETWHATYOUEAT. *Tweet what you eat!* Dec 2009. Acessado em Dezembro de 2009. Disponível em: <http://tweetwhatyoueat.com/>.

VIVOMETRICS. *VivoMetrics Web Site*. 2009. Acessado em Dezembro de 2009. Disponível em: <http://www.vivometrics.com/site/system.html>.

WEIGHTWATCHERS. *WeightWatchers.com - Official Site*. Dec 2009. Acessado em Dezembro de 2009. Disponível em: <http://www.weightwatchers.com/Index.aspx>.

WEIPPL, E.; HOLZINGER, A.; TJOA, A. M. Security aspects of ubiquitous computing in health care. *originalarbeiten*, p. 1–6, Jun 2006.

WEISER, M. The computer for the 21st century. *Scientific American*, p. 265–268, Mar 1991.

WHO. Full report - preventing chroninc diseases. p. 1–200, Sep 2005.

WHO. *Prevenção de doenças crônicas*. [S.l.], Oct 2005. 1-36 p.

WHO. *The impact of chronic disease in Brazil*. [S.l.], Oct 2009. 1-2 p.

WOOD, A. D. et al. Context-aware wireless sensor networks for assisted living and residential monitoring. *Ieee Network*, v. 22, n. 4, p. 26–33, Jan 2008.

ZIMMERMAN, T. Wireless networked digital devices: A new paradigm for computing and communication. *Ibm Syst J*, v. 38, n. 4, p. 566–574, Jan 1999.