UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE ELETRICIDADE

Dejailson Nascimento Pinheiro

MHNCS: Um middleware para o desenvolvimento de aplicações móveis cientes de contexto com requisitos de QoC

São Luís
2014

Dejailson Nascimento Pinheiro

MHNCS: Um middleware para o desenvolvimento de aplicações móveis cientes de contexto com requisitos de QoC

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Eletricidade da Universidade Federal do Maranhão como requisito obrigatório para a obtenção do grau de MESTRE em Engenharia de Eletricidade com área de concentração em Ciência da Computação.

Orientador: Francisco José da Silva e Silva

Prof. Doutor em Ciência da Computação

Universidade Federal do Maranhão

São Luís

2014

Pinheiro, Dejailson Nascimento

MHNCS: Um middleware para o desenvolvimento de aplicações móveis cientes de contexto com requisitos de QoC / Dejailson Nascimento Pinheiro. – São Luís, 2014.

117 f.

Orientador: Francisco José da Silva e Silva.

Impresso por computador (fotocópia).

Dissertação (Mestrado) – Universidade Federal do Maranhão, Programa de Pós-Graduação em Engenharia de Eletricidade. São Luís, 2014.

1.Computação Ubíqua. 2.Computação Ciente de Contexto. 3.Modelo *Publish/Subscribe*. I. Título.

CDU 004.75

MHNCS: UM MIDDLEWARE PARA O DESENVOLVIMENTO DE APLICAÇÕES MÓVEIS CIENTE DE CONTEXTO COM REQUISITOS DE QOC

Dejailson Nascimento Pinheiro

Dissertação aprovada em 06 de agosto de 2014.

Prof. Francisco José da Silva e Silva, Dr.
(Orientador)

Prof. Rafael Fernandes Lopes, Dr.
(Membro da Banca Examinadora)

Prof. Zair Abdelouahab, Ph. D.
(Membro da Banca Examinadora)

Dedico este trabalho aos meus pais, meus amigos e professores.

RESUMO

Redes Sociais Móveis (RSMs) são estruturas sociais em que seus membros relacionam-se em grupos e a interação é realizada através de tecnologias de informação e comunicação utilizando dispositivos portáteis com acesso a tecnologias de rede sem fio. Entre os muitos domínios de aplicação das RSMs, temos a área da saúde. O projeto *MobileHealthNet*, desenvolvido em parceria pela UFMA e PUC-Rio, tem por objetivo desenvolver um *middleware* que permita o acesso às redes sociais e facilite o desenvolvimento de serviços colaborativos para o setor da saúde, a troca de experiências e a comunicação entre pacientes e profissionais da saúde, além de uma melhor gestão dos recursos da saúde por órgãos governamentais. Um aspecto importante no desenvolvimento do *middleware* proposto pelo projeto *MobileHealthNet* é a infraestrutura necessária para a coleta, distribuição e processamento de dados de contexto.

Neste trabalho de mestrado é proposta uma infraestrutura de software incorporada ao *middleware MobileHealthNet* que permite a especificação, obtenção, validação e distribuição de dados de contexto, considerando requisitos de qualidade, tornando-os disponíveis a aplicações sensíveis ao contexto. A distribuição dos dados de contexto é baseado no modelo *publish/subscribe* centrado em dados, utilizando-se a especificação OMG-DDS.

Palavras-chaves: Computação Ubíqua, Computação Ciente de Contexto, Distribuição Centrado em Dados, Modelo *Publish/Subscribe*, Qualidade de Contexto.

ABSTRACT

Mobile Social Networks (MSNs) are social structures in which members relate in groups and interaction is accomplished through information and communication technologies using portable devices and wireless network technologies. Healthcare is one among the many possible areas of RSMs application. The MobileHealthNet project, developed in partnership by UFMA and PUC-Rio, aims to develop a middleware that allows access to social networks and facilitate the development of collaborative services targeting the health domain, the exchange of experiences and communication between patients and health professionals, as well as a better management of health resources by government agencies. An important aspect in the development of the MobileHealthNet middleware is the infrastructure necessary for the gathering, distribution and processing of context data.

In this master thesis we propose a software infrastructure incorporated to the MobileHealthNet middleware that allows the specification, acquisition, validation and distribution of context data, considering quality requirements, making them available to context-aware applications. The distribution of context data is based on a data-centric the publish/subscribe model, using the OMG-DDS specification.

Keywords: Ubiquitous Computing, Context-Aware Computing, Data-Centric Distribution, Publish/Subscribe model, Quality of Context.

AGRADECIMENTOS

Esta seção é dedicada a homenagear a todos que de uma forma ou de outra contribuíram para a elaboração deste trabalho. Vários colegas estiveram ao meu lado dando apoio, auxiliando e fazendo com que o caminho a ser percorrido ficasse cada vez mais próximo.

Agradeço à Deus em primeiro lugar e a minha família, que sempre me apoiou em momentos difíceis nesses dois anos de curso, principalmente minha mãe Maria, que me incentiva nos estudos desde criança.

Ao meu orientador, Professor Dr. Francisco José Silva e Silva, que me auxiliou no desenvolvimento deste trabalho, apoiou em vários momentos de decisões não somente do trabalho, mas em várias outras questões. Pela paciência na correção dessa dissertação e esclarecimento de dúvidas. Mas agradeço principalmente por ele ter acreditado em mim, desde o início, ao me aceitar no Mestrado como seu aluno. Aos outros professores do PPGEE, que sempre me deram apoio e incentivo, aos meus colegas de trabalho do SENAI.

Agradeço a equipe do Laboratório de Sistemas Distribuídos (LSD) da UFMA, com os quais compartilhamos conhecimentos na área de pesquisa, em Computação Ciente de Contexto. Em especial, agradeço a Ariel e Berto, que contribuíram efetivamente com este trabalho.

O sucesso é construído à noite!

Durante o dia você faz o que todos fazem. Mas, para obter um resultado diferente da maioria, você tem que ser especial. Se fizer igual a todo mundo, obterá os mesmo resultados.

Roberto Shinyashiki

Lista de Figuras

Lista de Tabelas

Lista de Siglas

CDS	*Context Distribution Service*.
CES	*Context Event Service*.
CMS	*Context Management Service*.
DCPS	*Data-Centric Publish Subscribe*.
DDMS	*Dalvik Debug Monitor Server*.
DDS	*Distributed Data Service*.
EBNF	*Extended Backus-Naur Form*.
GPS	*Global Positioning System*.
H-SAUDE	*Heath Support in Aware and Ubiquitous Domestic Environments*.
HUPD Care	Hospital Presidente Dutra *Care*.
HUUFMA	Hospital Universitário da UFMA.
MHNCS	***M**obile**H**ealth**N**et **C**ontext **S**ervice*.
OMG	*Object Management Group*.
QoC	Qualidade de Contexto.
QoD	*Qualidade de Dispositivo*.
QoS	Qualidade de Serviço.
RSM	Redes Social Móvel.
RSMs	Redes Sociais Móveis.

SOAP	*Simple Object Acess Protocol*.
XML	*eXtensible Markup Language*.
XMPP	*Extensible Messaging and Presence Protocol*.
XSD	*XML Schema Definition*.

Sumário

1 INTRODUÇÃO

A computação ubíqua é uma forma de computação em que o processamento está espalhado no ambiente através de vários dispositivos, que executam tarefas bem definidas dependendo de sua natureza, interligados de forma que essa estrutura torna-se invisível para o usuário. Aplicações ubíqua executam em ambientes instrumentados com sensores, geralmente dotados de interfaces de redes sem fio, tais como dispositivos, aplicações e serviços que são integrados de forma transparente e cooperam para atender aos objetivos da aplicação. Essa categoria de aplicação caracteriza-se por constantes mudanças em seu estado de execução, geradas pelo ambiente altamente dinâmico em que executam [70].

Nos últimos anos é possível verificar um rápido crescimento na área da computação ubíqua [46], a qual tem sido cada vez mais inserida da sociedade. Dispositivos móveis fornecem conectividade e permitem acesso, processamento e compartilhamento de informação a qualquer tempo e em qualquer lugar, provendo ubiquidade de acesso a recursos computacionais. Eles também estão se tornando cada vez mais baratos e provendo mais recursos como, por exemplo, maior poder de armazenamento e processamento, múltiplas interfaces de rede, *Global Positioning System* (GPS), e integrando uma variedade de sensores, como sensor de proximidade, magnetômetro e acelerômetro. Dessa forma, dispositivos móveis vem permitindo a execução de aplicações cada vez mais sofisticadas e com capacidade de identificar alguns aspectos do contexto do usuário ou até sua atividade/situação em um dado momento.

Uma classe de aplicação que tem ganho destaque em dispositivos móveis são as redes sociais. Conceitualmente, Rede Social é uma estrutura de entidades conectadas umas às outras através de um ou mais tipos específicos de interdependências, tais como amizade, parentesco, interesse em comum, troca financeira, empatia, ou relações de crenças, conhecimento [68]. Estas entidades podem ser indivíduos, organizações ou sistemas que estão relacionados em grupos e cuja interação é possibilitada através de tecnologias da informação e comunicação. As Redes Sociais Móveis (RSMs) [31] ou Redes Sociais Pervasivas(Ubíqua) compreendem

uma subclasse das redes sociais, na qual os usuários utilizam dispositivos móveis com tecnologias de comunicação sem fio para acessar conteúdos das redes sociais. Assim, usuários móveis podem acessar (ler), publicar (escrever ou inserir), compartilhar (retransmitir ou divulgar) ou deixar disponíveis (permitir requisições de acesso originadas por contatos) conteúdos criados por si mesmos, ou obtido por de sensores do dispositivo móvel, para a exploração de suas relações sociais.

Neste cenário, aplicações sociais móveis são caracterizadas por adicionar informações de contexto às redes sociais, uma vez que dispositivos móveis obtêm dados físicos do ambiente e, assim, é possível combinar dados de contexto e inferir a situação dos usuários. Então, uma vez que dispositivos móveis proveem ubiquidade de acesso, RSMs possibilitam interações sociais entre entidades de forma a melhorar relações existentes ou criar novas a partir de interesses em comum, tais como lugares visitados e frequentados, esportes praticados, preferências musicais, mensagens publicadas, ou até mesmo a situação de saúde em que os usuários se encontram (RSMs aplicadas à saúde). Dentro da linha de pesquisa de RSMs aplicadas à saúde, está sendo desenvolvido o projeto MobileHealthNet [59] [58], uma parceria entre o Laboratório de Sistemas Distribuídos da Universidade Federal do Maranhão (UFMA) e o *Laboratory for Advanced Collaboration* da Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio). Este projeto tem por objetivo geral avançar o estado da arte em sistemas de *middleware* para RSMs voltadas para a área da saúde.

Um aspecto importante no desenvolvimento do *middleware* proposto pelo projeto MobileHealthNet é a infraestrutura necessária para a coleta, distribuição e processamento de dados de contexto. Contexto pode ser definido como qualquer informação que pode ser usada para caracterizar a situação de uma entidade (pessoa, local ou objeto) que é considerada relevante para a interação entre o usuário e a aplicação, incluindo o próprio usuário e a aplicação [19]. Segundo Schilit e Theimer [50], aplicações sensíveis ao contexto podem ser construídas para retornar respostas úteis por meio da observação e reação ao ambiente. Isso é, a sensibilidade ao contexto refere-se à capacidade de uma aplicação de perceber características de seu ambiente, e é um requisito primordial para permitir a adaptação em resposta às mudanças ambientais.

Ao se construir e executar aplicações ubíqua sensíveis ao contexto, há uma série de funcionalidades que devem ser providas, envolvendo desde a aquisição de

informações de contexto, a partir de um conjunto de fontes heterogêneas e distribuídas, até a representação dessas informações, seu processamento, armazenamento, e a realização de inferências para seu uso em tomadas de decisão. Devida a essa complexidade, em vez de deixar essas funcionalidades a cargo da aplicação, as incorporando ao seu código, são utilizadas como infraestruturas subjacentes as plataformas ou *middleware* de provisão de contexto [19] [25] [47]. O foco deste trabalho de mestrado é o provimento da infraestrutura de suporte ao desenvolvimento de aplicações sensíveis ao contexto a ser incorporado ao *middleware* do projeto MobileHealthNet.

Aplicações ubíqua aplicadas em diversas áreas estão impondo novos requisitos a obtenção e distribuição de dados de contexto. Em alguns casos, além de obter e distribuir as informações, é necessário fazer com que elas cheguem até as aplicações com um certo nível de qualidade. A qualidade das informações utilizadas pelas as aplicações sensíveis a contexto tem um impacto significativo para essas aplicações, principalmente quando aplicado à área da saúde, pois as tomadas de decisões dos profissionais da saúde baseadas nesses dados são cruciais, atingindo diretamente o estado de saúde do paciente. A Qualidade de Contexto (QoC) descreve a qualidade de uma determinada informação de contexto e desta forma pode auxiliar a aplicação a estimar o comportamento de um serviço sensível ao contexto ou também pode servir como um indicador para a seleção de uma fonte de contexto mais adequado.

1.1 Objetivos

Neste trabalho, tem-se como objetivo geral o desenvolvimento de um *middleware* com suporte à especificação, solicitação, obtenção, validação e distribuição de dados de contexto, considerando requisitos de QoC, tornando-os disponíveis a aplicações sensíveis ao contexto tanto locais (que executem no dispositivo no qual a coleta dos dados de contexto é realizada) quanto remotas (que executem em qualquer outro nó da rede).

1.2 Objetivos Específicos

- Definição de uma linguagem que facilite a definição de requisitos de dados de contexto possibilitando a definição de parâmetros de Qualidade de Contexto (QoC) por parte do desenvolvedor de aplicações cientes de contexto;
- Desenvolvimento de mecanismos para o gerenciamento e a coleta de dados de contexto em ambientes móveis de larga escala;
- Definição de um modelo centrado em dados para a distribuição de dados de contexto;
- Desenvolvimento de um sistema de avaliação e de notificação de eventos de contexto;
- Avaliação da infraestrutura de contexto a ser desenvolvida .

1.3 Organização do Trabalho

O restante deste trabalho está organizado da seguinte forma:

- No Capítulo 2, é apresentado uma fundamentação teórica sobre redes sociais móveis, destacando o *middleware* MobileHealthNet e sua respectiva arquitetura;
- No Capítulo 3, é exibida uma fundamentação teórica abordando o tema Computação Ciente de Contexto, o qual é conhecimento necessário para o entendimento deste trabalho;
- O Capítulo 4 descreve a fundamentação teórica sobre a arquitetura centrada em dados, tendo como base o paradigma de comunicação *Publish/Subscribe*;
- O Capítulo 5 descreve uma solução para a distribuição de dados de contexto desenvolvida para o *middleware* MobileHealthNet, tendo como base o modelo *Data-Centric Publish/Subscribe*;
- No Capítulo 6 são descritos os trabalhos relacionados ao problema tratado nesta pesquisa de mestrado e faz um comparativo entre eles;

- O Capítulo 7 apresenta os resultados dos experimentos realizados sobre a infraestrutura de distribuição de dados desenvolvida para o MobileHealthNet;

- Por fim, o Capítulo 8 aborda as conclusões extraídas deste trabalho de mestrado, além de apresentar alguns trabalhos futuros que podem dar continuidade a esta pesquisa.

2 REDES SOCIAIS MÓVEIS

Este capítulo faz uma apresentação das RSMs, inicialmente descrevendo conceitos de redes sociais e posteriormente comentando sobre o papel das informações de contexto em RSMs e também as arquiteturas de RSMs. Por fim, o projeto MobileHealthNet é detalhado.

2.1 Considerações Iniciais

Redes Sociais Online podem ser definidas como um conjunto de serviços baseados na web que permitem a indivíduos: (1) construir um perfil público ou semi-público dentro do sistema; (2) manter uma lista de contatos[1], se comunicar com os membros dessa lista e estabelecer elos para futuras interações; (3) consultar sua lista de contatos e de outros amigos e conhecidos dentro do sistema [11]. A semântica, a natureza e o tipo de relacionamento criado com esses elos pode variar entre sistemas, mas são genericamente representadas por um grafo, juntamente com os perfis dos usuários, chamado de Grafo Social. No grafo social, os vértices expressam o perfis dos usuários e as arestas os relacionamentos entre eles.

Como exemplo de um grafo social descrito em [60], quando um usuário (João) possui um elo com outro (Maria), existe uma aresta unindo o vértice A (João) ao vértice B (Maria), como ilustrado na Figura 2.1. Neste caso, o elo entre os usuários é mútuo, como ocorre, por exemplo, no Facebook. Porém, dependendo da rede social, essa aresta pode possuir uma orientação (direção), como é visto no grafo social B na mesma figura. Este é o caso de redes sociais como Twitter e Instagram, onde é utilizado o recurso "seguir". Como visto na aresta B, João segue Maria para ter acesso a conteúdos publicados por ela. No entanto, observa-se que Maria não segue João e, caso o perfil dele seja privado, ela não tem acesso a conteúdos publicados por ele. A aresta C também possui orientação, mas neste caso os dois usuários (Pedro e Ricardo) seguem um ao outro.

[1]Os termos elo e contato são utilizados como sinônimos durante o texto.

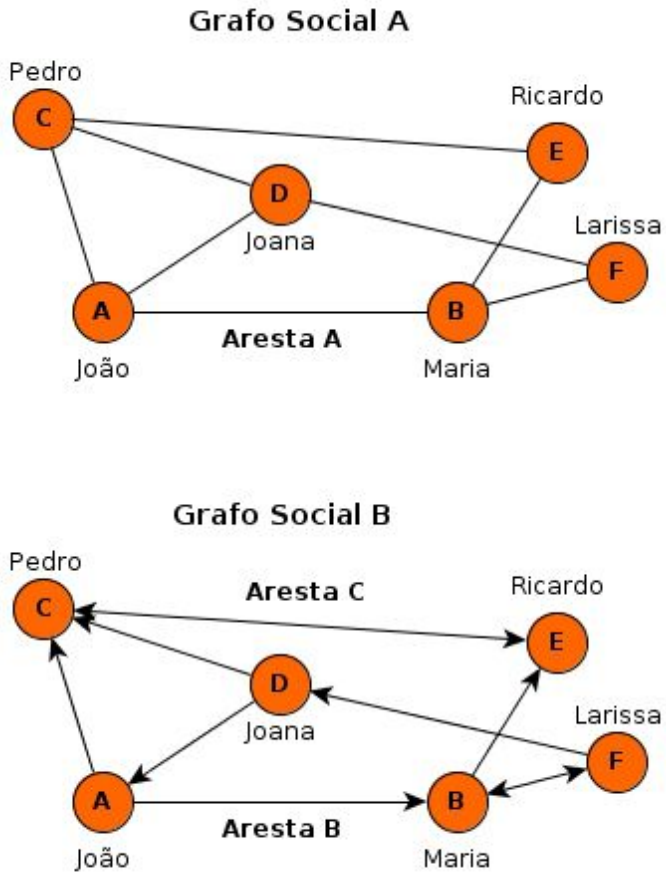

Figura 2.1: Grafo Social.

Usuários de redes sociais tipicamente geram uma grande quantidade de conteúdos, tais como fotos, áudio e vídeos. A representação de redes sociais através de um grafo facilita o entendimento da maneira pela qual esses conteúdos são disseminados na rede, percorrendo arestas e passando por vértices, sendo visualizados por outros usuários.

A exibição pública dos contatos é um componente intrínseco das redes sociais. A lista de contatos contém links para o perfil de cada usuário, permitindo navegar pelas listas de contatos, e assim percorrer o grafo social. Na maioria das redes sociais, a lista de contatos permanece visível para quem está autorizado a visualizar o perfil do usuário, embora haja exceções. Por exemplo, no LinkedIn é possível aos usuários optar por não exibir seus contatos conhecidos.

A maioria das redes sociais também inclui mecanismos para deixar mensagens nos perfis de contatos, os chamados "comentários". Porém, cada rede social usa um nome diferente para este recurso. Além disso, redes sociais muitas vezes

possuem o recurso de mensagens privadas. Embora seja comum a disponibilização de ambas as formas de comunicação (pública e privada), elas não estão presentes em todas as redes sociais.

RSMs compreendem um subconjunto das redes sociais online. RSMs podem ser vistas como uma combinação de três áreas do conhecimento: Redes Sociais, Computação Móvel e Ciência de Contexto [61], uma extensão do conceito formalizado em [31]. Essa visão de RSMs é ilustrada na Figura 2.2.

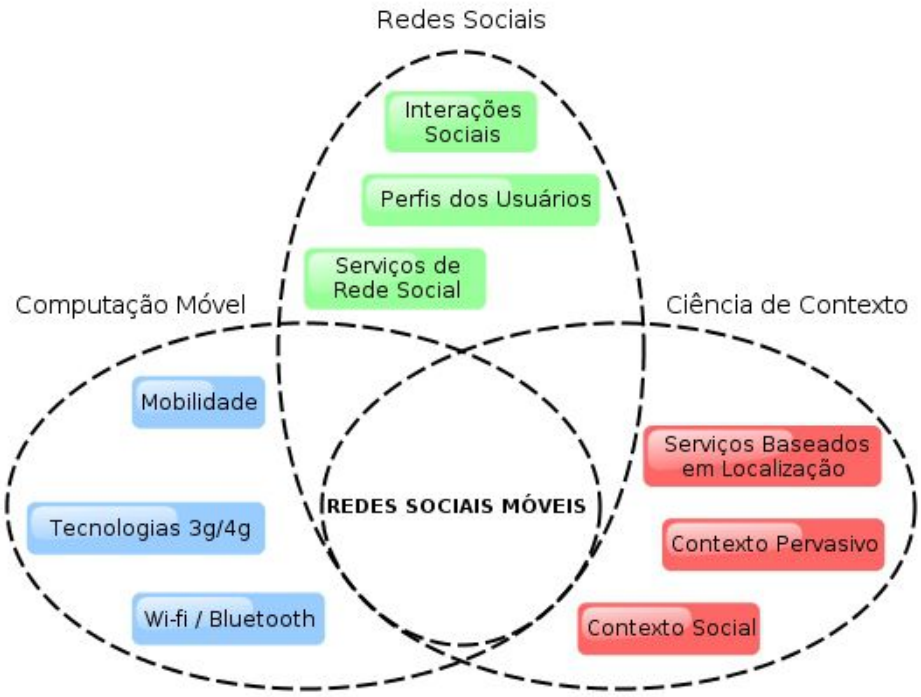

Figura 2.2: Definição de RSMs.

Fonte Teles et al. [61]

Redes sociais provem funcionalidades para criar perfis que representam entidades, as quais relacionam-se socialmente trocando informações. Como visto, estas entidades podem ser indivíduos, organizações ou mesmo sistemas. A Computação Móvel possibilita os usuários estarem sempre online, devido ao suporte de mobilidade provido pelos dispositivos portáteis e a ubiquidade da conectividade sem fio. Ciência de Contexto adapta as funcionalidades das redes sociais e suas aplicações, oferecendo recursos de acordo com informações de contexto, o que será visto em detalhes a seguir.

2.1.1 Inserção de Contexto em RSMs

A principal motivação para a inserção de contexto é formar RSMs nas quais a interação e a percepção mútua entre os usuários pode ser induzida ou aprimorada através do compartilhamento de informações de contexto. Um usuário em uma festa, por exemplo, pode compartilhar a sua localização com seus contatos das redes sociais e, então, é possível que encontre algum conhecido que compartilhou a informação de também estar nessa mesma festa e, assim, eles podem se encontrar.

Dados gerados a partir de sensores ubíquo podem ser usados individualmente ou de forma combinada para inferir a situação do usuário. Essa situação pode ser, por exemplo, a (in)disponibilidade do usuário para realizar alguma atividade, seu estado de saúde ou o lugar em que se encontra (por exemplo, restaurante, residência ou local de trabalho). Portanto, informações de contexto ubíquo são bastante úteis para aplicações de RSMs.

A inferência da situação do usuário a partir de vários sensores pode prover uma variedade de informações a serem usadas por aplicações de RSMs para aumentar o nível de colaboração entre seus usuários. Por exemplo, informações do GPS podem revelar que o dispositivo, e o seu usuário, estão em movimento e, a partir disso, a aplicação pode fornecer serviços de alerta para notificar amigos do usuário se ele está ocupado para receber notificações, conexões ou chamadas. Com isso, usuários de RSMs podem ficar mais cientes da situação de seus amigos na sua rede social, o que leva a uma maior integração entre os mundos real e virtual.

Um exemplo dessa integração entre mundos real e virtual é o Projeto *Touch Me Wear*. Neste projeto, as informações de contexto são coletadas através de dispositivos vestíveis, em que o usuário utiliza uma camisa com sensores de toque e dispositivo *bluetooth* e, dessa forma são obtidos, respectivamente, os contatos físicos entre usuários (por exemplo, toque ou abraço) e informações de quem está na vizinhança (usuários próximos). A partir disso, quando dois usuários tocam-se fisicamente, abraçam-se ou estão próximos, essa informação é publicada em seus perfis, na rede social Facebook, com informações de quem o usuário tocou/abraçou ou esteve próximo e quando isso ocorreu. A tecnologia utilizada por esse projeto é chamada de *Body Sensor Network*, na qual sensores monitoram atividades fisiológicas e ações de humanos [15].

Nesse cenário surgem as Redes Geo-sociais, as quais proveem serviços cientes de contexto que agregam a localização de usuários a conteúdos [63]. Por exemplo, a publicação de uma foto pode agregar informação sobre a localização na qual a foto foi tirada. Essa classe de RSMs combina as tecnologias de informação geográficas e serviços de redes sociais para ajudar pessoas a estabelecer relações sociais mais facilmente [29]. Essa facilidade surge devido ao interesse das pessoas em algo relacionado a uma localização específica, por exemplo, usuários que estudam na UFMA e querem fazer amizades com outros que estudam nessa mesma universidade. Informações de localização são usadas por várias aplicações de RSMs, tais como o Google Latitude[2] e o Foursquare.

A inserção de informações de contexto às RSMs permite o estabelecimento dinâmico de elos e a troca de conteúdos entre usuários. Este recurso é chamado de Comunidades Dinâmicas (ou Grupos Dinâmicos) [38]. Comunidades Dinâmicas correspondem a grupos de usuários que são automaticamente criados baseados em (i) interesses comuns derivados dos perfis de usuários, (ii) inferência de interesses comuns baseada no histórico de atividades do usuário, (iii) proximidade física de usuários ou localização. Como exemplo, estudantes de um mesmo curso trocando informações sobre uma prova antes dela ser aplicada ou pessoas que estejam assistindo a um mesmo espetáculo (por exemplo, pessoas que *Tweetam* [3] sobre um show ou um acontecimento) sendo presenciado conjuntamente. Neste caso, a rede social se torna mais dinâmica, uma vez que usuários podem entrar e sair das comunidades, pois eles podem se deslocar de uma lugar para outro ou mudar seus interesses.

2.2 Arquiteturas de RSMs

Na literatura, embora seja possível encontrar diversas arquiteturas para RSMs que permitem o estabelecimento de várias comunidades de usuários móveis e suas interações, sistemas de RSMs podem ser classificados em dois principais grupos: centralizado e distribuído, como ilustrados na Figura 2.3.

Em uma arquitetura centralizada, os dados estão centralizados em um ou mais servidores. Esses servidores são responsáveis pelo gerenciamento e entrega

[2]https://www.google.com/latitude/

[3]Ato de fazer um comentário ou postagem na rede social Twitter.

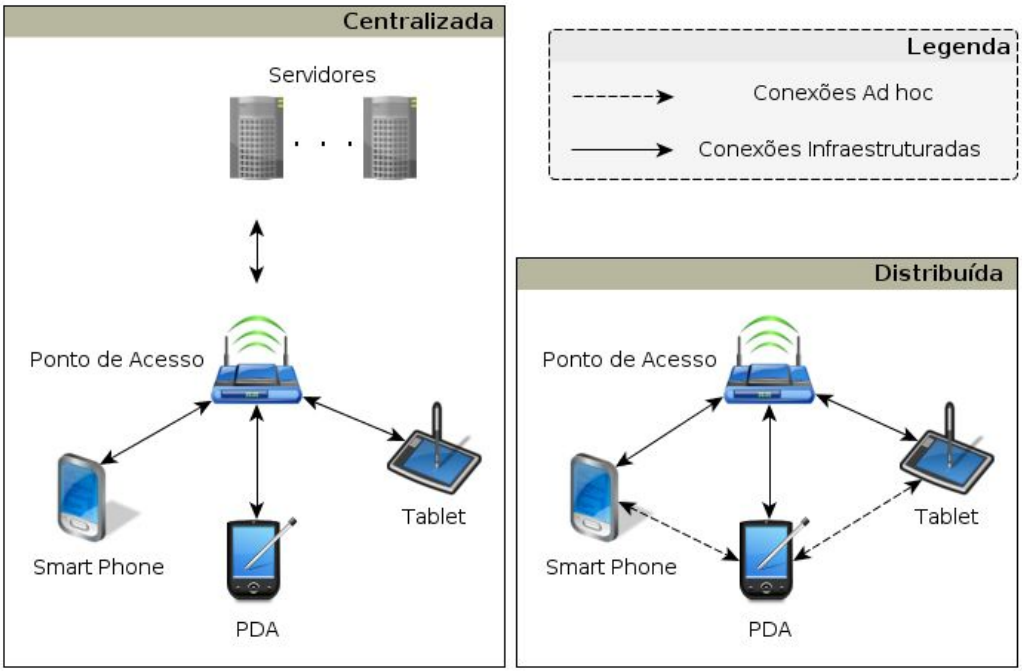

Figura 2.3: Arquiteturas de RSMs

Fonte: Teles et al. [61]

dos dados aos usuários móveis. Esses dados correspondem a informação dos perfis dos usuários, grupos, informações de contexto, murais, notificações, dentre outras. Usuários móveis são representados por aplicações móveis instaladas/armazenadas nos dispositivos móveis. Nessa arquitetura, toda comunicação é mediada pelos servidores. Dessa forma, os conteúdos são necessariamente acessados através dos servidores, e somente a partir deles, através de conexões estabelecidas pelas aplicações móveis.

Em uma arquitetura distribuída, usuários móveis comunicam-se diretamente, sem a necessidade do uso de servidores para mediar suas interações. Como visto na Figura 2.3, conexões entre dispositivos de usuários podem ser estabelecidas através de uma infraestrutura de rede usando pontos de acesso ou por meio de conexões *ad-hoc*. Nessa arquitetura, usuários podem se comunicar e compartilhar conteúdo sem acesso a Internet, com o mínimo de infraestrutura de rede. Além disso, conhecimento adquirido a partir das relações sociais entre usuários pode ser usado para criar um melhor roteamento e protocolos de segurança. Por exemplo, a frequência na qual usuários encontram-se fisicamente pode representar um maior nível de confiança para uma rota, pois a probabilidade dos usuários se encontrarem e rotearem um dado é maior [33].

Outra forma que componentes de uma Redes Social Móvel (RSM) podem ser organizados é através de uma arquitetura híbrida. Essa arquitetura combina as abordagens centralizadas e distribuídas por permitir que usuários móveis acessem e compartilhem dados através de servidores (centralizada), enquanto outros usuários (geralmente co-localizados ou próximos uns dos outros) estabeleçam conexões diretas uns com os outros sem a necessidade de um servidor (distribuída). Dessa forma, essa arquitetura pode aproveitar os benefícios oferecidos por ambas arquiteturas.

Essas arquiteturas implicam em diferentes formas de comunicação para obtenção de acesso a conteúdos das RSMs e entre os usuários, o que modifica a maneira deles interagirem, visto que alguns casos há a necessidade de uma conexão à Internet e, em outros, basta que os usuários estejam próximos fisicamente. Por exemplo, em uma arquitetura centralizada é apenas necessário que o usuário tenha uma conexão com a Internet para acessar qualquer conteúdo, diferentemente de uma arquitetura distribuída, em que em muitos casos os usuários precisam estar próximos fisicamente de algum ponto de acesso, ou ainda estarem próximos de algum outro usuário que tenha conexão com a Internet. Neste último caso, um usuário teria conexão com a Internet e serviria de ponte para um segundo usuário obter acesso aos conteúdos da RSMs.

A escolha da arquitetura para construção de uma RSM tem grande impacto nos serviços que são disponíveis, pois a utilização de uma arquitetura pode não ser adequada para atender os requisitos de uma determinada aplicação. Além disso, a arquitetura é decisiva para a escolha dos algoritmos que serão adotados na construção do *software*.

2.3 O Projeto MobileHealthNet

O projeto MobileHealthNet tem por objetivo avançar o estado da arte em sistemas de *middleware* para RSMs voltadas para a área da saúde. Esse projeto conta com apoio institucional do Hospital Universitário da UFMA (HUUFMA). Em particular, duas unidades do HUUFMA estão diretamente envolvidas com o desenvolvimento do projeto: o Programa de Assistência a Pacientes Asmáticos (PAPA)

e a Liga Acadêmica da Dor (LAD), esta última especializada no tratamento de pacientes que sofrem com dores crônicas, independentemente de sua etiologia[4].

O projeto MobileHealthNet está focado em comunidades carentes e distantes, onde o custo do deslocamento de pacientes para os postos de atendimento é alto devido a suas condições de acesso. Dessa forma, pode-se explorar o uso de RSMs para prover o acompanhamento remoto do tratamento desses pacientes, permitindo que eles mantenham contato com profissionais da saúde além dos encontros presenciais. Além disso, o MobileHealthNet pode ser usado para educação de pacientes e treinamentos de profissionais, pela inserção de conteúdos educativos pertinentes, como o correto uso de medicamentos. O MobileHealthNet proporciona também um ambiente de interação entre os profissionais, fomentando discussões sobre os mais diversos temas relacionados ao tratamento de pacientes. Finalmente, ele pode ainda ser utilizado para fomentar comunidades de pacientes que possuam condições de saúde ou realizem tratamentos semelhantes, fomentando empatia e troca de experiências.

O *middleware* MobileHealthNet (*middleware* resultante do projeto) disponibiliza uma infraestrutura de software para a criação de novos serviços e aplicações sociais móveis aplicadas ao domínio da saúde. O *middleware* e as aplicações são desenvolvidos tendo como base um conjunto de requisitos elicitados a partir de diversas reuniões com os profissionais da saúde envolvidos no projeto. Essas reuniões tem como objetivo o conhecimento do domínio e das principais necessidades de ambos os núcleos (PAPA e LAD). Dessa forma, foram aplicadas técnicas de coleta de requisitos, como entrevistas e *brainstorms*, em conjunto com os profissionais da saúde, a fim de se obter os principais requisitos funcionais e não funcionais das aplicações.

Entre os benefícios esperados com o MobileHealthNet, destaca-se um melhor fluxo de informação e maior colaboração entre profissionais dos diversos níveis de atendimento à saúde; melhorias nos níveis de comprometimento e informação do paciente, o que contribui com o processo terapêutico; melhor gestão de pacientes portadores de doenças crônicas, levando à diminuição da ocorrência de complicações; melhoria na qualidade da tomada de decisão relativa ao tratamento por parte dos pacientes; melhor suporte emocional aos pacientes e redução de custos do sistema de

[4]Estudo, análise e pesquisa sobre as causas das doenças

atendimento à saúde, diminuindo-se a necessidade de deslocamentos e a ocorrência de complicações no tratamento de pacientes.

A arquitetura do *middleware* leva em consideração os seguintes requisitos: (i) componentes de software para dispositivos móveis devem apresentar baixo consumo de recursos, de maneira que possam ser executados em uma grande variedade de equipamentos, incluindo aqueles considerados de baixo custo, já que os resultados do projeto devem ser aplicáveis a comunidades carentes; (ii) deve-se prover suporte a diversos tipos de comunicação sem fio (em especial sistemas celulares e redes locais sem fio), de forma a se atingir uma ampla área de cobertura, minimizando-se custos de comunicação sempre que possível; (iii) deve-se prover meios para a construção e compartilhamento de conteúdo de forma colaborativa por seus usuários; (iv) os usuários devem ter a liberdade de criar e participar de diversos grupos de acordo com seu interesse, fomentando-se assim a criação de comunidades dinâmicas que, no entanto, devem respeitar critérios de privacidade; (v) disponibilizar meios para a interação de forma síncrona e assíncrona entre participantes das comunidades; (vi) deve-se prover mecanismos que permitam a comunicação em tempo real e com suporte a Qualidade de Serviço (QoS), de forma a habilitar a notificação de eventos prioritários e a transmissão de dados de monitoração remota de pacientes; (vii) fornecer um modelo de segurança e privacidade capaz de proteger dados sensíveis de saúde; e (viii) aplicações devem apresentar interfaces intuitivas e com alta usabilidade, considerando a idade, nível educacional e problemas físicos (por exemplo, surdez ou cegueira) dos potenciais usuários. A arquitetura do estado atual do *middleware* é organizada em quatro camadas, ilustrada na Figura 2.4 e descrita a seguir.

A camada *Communication Layer* [7] é responsável por prover os mecanismos de comunicação entre os componentes que compõem as camadas superiores. Ela utiliza como mecanismo de comunicação uma versão adaptada do SDDL [18] para o *middleware* MobileHealthNet , a qual possibilita o desenvolvimento centrado em dados usando tópicos nos dispositivos móveis. Resumidamente, o SDDL utiliza o protocolo MR-UDP[5] para a comunicação remota entre os dispositivos móveis e o *gateway*. O *gateway* é um servidor que concentra conexões de vários dispositivos móveis remotos para acesso ao domínio local (*Data Global Space DDS*). No domínio local é utilizado

[5] `http://lac-rio.com/mr-udp/`

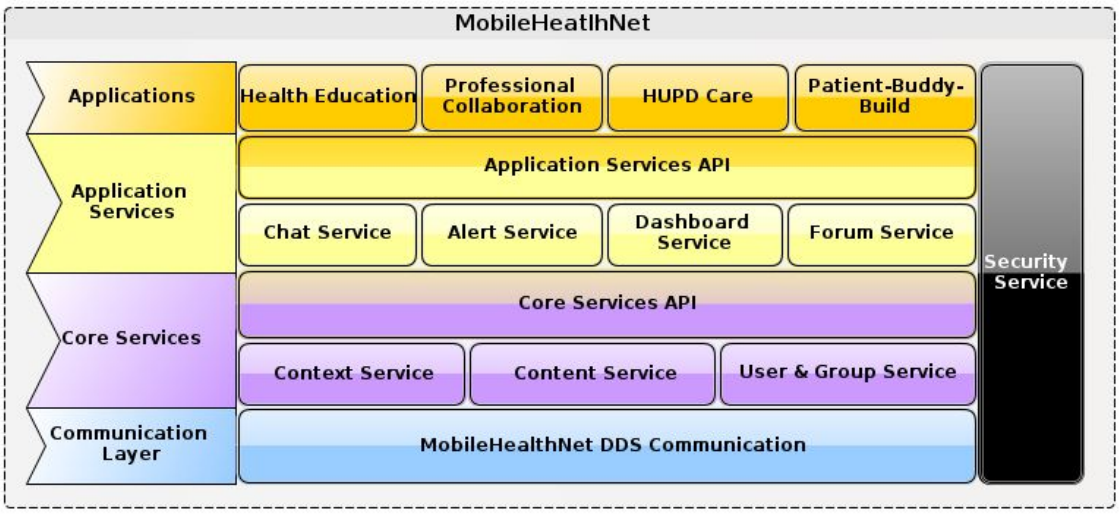

Figura 2.4: Arquitetura do MobileHealthNet.

Fonte: Teles et al. [59]

uma comunicação *Object Management Group* (OMG) *Distributed Data Service* (DDS)[6], uma especificação para comunicação *publish/subscribe* que possibilita a atribuição de políticas de QoS visando a distribuição das informações em sistemas distribuídos de tempo real. O DDS foi projetado para prover escalabilidade, portabilidade e interoperabilidade entre plataformas. No DDS os dados são trafegados em formato de tópicos, em que o dado é publicado no domínio DDS através de um determinado tópico e os interessados neste dado se subscrevem a este tópico, recebendo notificações sempre que um novo dado estiver disponível.

A camada *Core Services* disponibiliza serviços básicos a serem utilizados por serviços específicos e aplicações. O *Context Service*, que refere-se ao ***M**obile**H**ealth**N**et **C**ontext **S**ervice* (MHNCS) responsável pela especificação, solicitação, obtenção, validação distribuição dos dados de contexto. O *Content Service* tem como principal tarefa o compartilhamento de mídias (por exemplo, textos, fotos, áudios, filmes) entre usuários da rede social. O *User & Group Service* gerencia a criação de contas de usuários e grupos, bem como o gerenciamento do grafo social.

A camada *Application Services* disponibiliza serviços típicos de redes sociais, como serviços de publicação de mensagens em murais, fórum e chat, além de um serviço de notificações para os usuários. A camada *Security Services* [24] é transversal

[6]Data Distribution Service for Real-Time Systems Specification em:<http://www.omg.org/spec/DDS/1.2/PDF/>. acesso em janeiro de 2014

a todo o código gerado no MobileHealthNet e disponibiliza os componentes responsáveis pelos mecanismos de segurança e privacidade. Importante destacar que esta camada do *middleware* é desenvolvida em conformidade com o Manual de Certificação para Sistemas de Registro Eletrônico em Saúde (S-RES) da Sociedade Brasileira de Informática em Saúde (SBIS).

A camada *Applications* refere-se às aplicações previstas no projeto. Quatro aplicações específicas para a área da saúde estão previstas para serem desenvolvidas nesta etapa inicial: (i) HealthEducation possui ferramentas para o compartilhamento de arquivos multimídia e seu objetivo é aprimorar a educação de pacientes e profissionais da saúde através de mídias com conteúdo educacional; (ii) Professional Collaboration visa diminuir a distância entre os profissionais da saúde, através de recursos como *chat* multiusuário, fórum de discussão e serviço de notificação que permita informar sobre a urgência de se obter o resultado de um exame, discutir a respeito do tratamento e acompanhamento de um determinado paciente; (iii) Hospital Presidente Dutra *Care* (HUPD Care) tem por objetivo explorar os conceitos das RSMs para promover a assistência prestada por especialistas responsáveis pelo atendimento de alta complexidade a profissionais da atenção básica a saúde no atendimento a casos específicos e; (iv) Patient-Buddy-Build [4] que consiste de uma ferramenta para a geração de aplicações móveis, contendo um questionário customizado, para o acompanhamento à distância de pacientes com doenças crônicas.

2.4 Considerações Finais

Esse capítulo apresentou uma fundamentação sobre o cenário em que este trabalho está inserindo, as RSMs. Nele foram vistos os conceitos fundamentais de redes sociais e mais especificamente de RSMs, mostrando como as informações de contexto são utilizadas nesta classe de sistemas. Os vários tipos de arquiteturas que podem ser utilizadas na implementação de RSMs também foram mostrados. Ao final, o projeto MobileHealthNet foi detalhado.

3 COMPUTAÇÃO CIENTE DE CONTEXTO

A Computação Ciente de Contexto refere-se à capacidade de um sistema computacional perceber características do meio ambiente que sejam de seu interesse [49]. Aplicações ciente ao contexto conhecem o ambiente no qual estão sendo utilizadas e tomam decisões de acordo com mudanças no seu próprio ambiente. Tais aplicações reagem a ações executadas por outras entidades, podendo essas ser pessoas, objetos ou até mesmo outros sistemas, que modifiquem o ambiente. Essas aplicações, de um modo geral, tomam ciência de modificações que venham a acontecer no ambiente. A tais modificações denomina-se alterações nas informações de contexto. O contexto pode ser tudo que está ao redor de um sistema em questão, tudo que ocorre em um determinado ambiente, tudo que é relevante às aplicações. Neste capítulo faz-se uma apresentação sobre a ciência de contexto, inicialmente descrevendo conceitos introdutórios e posteriormente abordando os tipos de modelo para representar as informações de contexto e aspectos relacionados à Qualidade de Contexto QoC.

3.1 Ciência de Contexto

Anind K. Dey [20] define contexto como sendo qualquer informação que pode ser usada para caracterizar a situação de uma entidade. Uma entidade pode ser uma pessoa, lugar ou objeto que seja considerado relevante na interação do usuário com a aplicação. Sistemas sensíveis ao contexto estão habilitados a adaptar suas funcionalidades e comportamento de acordo com o contexto atual do usuário sem sua explícita intervenção.

Bolchini et al. [9] descreve as informações de contexto como sendo um conjunto de variáveis que podem ser de interesse para uma entidade e podem influenciar em suas ações. Por meio desse tipo de informação é possível o desenvolvimento de sistemas computacionais que podem se reconfigurar ou adaptar a uma determinada situação ou recomendar ações a partir de análises de informações coletadas do ambiente.

A utilização de informações de contexto permite que desenvolvedores de sistemas possam enriquecer a usabilidade de sua aplicação e, assim, o sistema sensível ao contexto pode reagir a determinadas situações sem a necessidade da interação com o usuário. Por exemplo, caso o sistema detecte que o nível de bateria está baixo, ele pode realizar ações para economizá-la, seja reduzindo a luminosidade do visor do dispositivo ou desativando as interfaces de rede ou sensores que não estão em uso no momento.

Para Chen e Kotz [14] as informações de contexto podem ser classificadas em quatro tipos:

- **Contexto Físico**: informações sobre o mundo real, obtidas por meio de sensores. Por exemplo, sensor de luminosidade, sensor de ruido, temperatura, luminosidade e localização;
- **Contexto Computacional**: informações sobre um sistema computacional, como os seus recursos e características. Por exemplo, nível de bateria, consumo de memória ou processamento e conexões de rede disponíveis;
- **Contexto do Usuário**: informações que caracterizam o usuário, tais como: estado emocional, localização e atividade atual;
- **Contexto de Tempo**: informações relacionadas ao tempo de uma atividade real ou virtual. Está relacionada a dimensão do tempo, como por exemplo, hora do dia, dia da semana, mês, ano ou uma estação climática no ano.

Uma outra classificação mais atual é proposta por Emmanouilidis et al. [21] a qual possui cinco tipos de informações de contexto. A Figura 3.1 ilustra essa classificação. Essa classificação mantém uma relação com a proposta de Chen e Kotz [14] descrita anteriormente: usuário (*User*) equivale ao contexto do usuário, sistema (*System*) equivale ao contexto computacional e, por fim, ambiente (*Environment*) equivale a junção de contexto físico com contexto de tempo. Além desses, outros dois tipos de contexto são propostos por essa classificação, são eles: Social e Serviço (*Service*).

O termo Contexto Social é utilizado para caracterizar as possíveis formas de relacionamento e de interações entre pessoas, intermediadas ou não por alguma

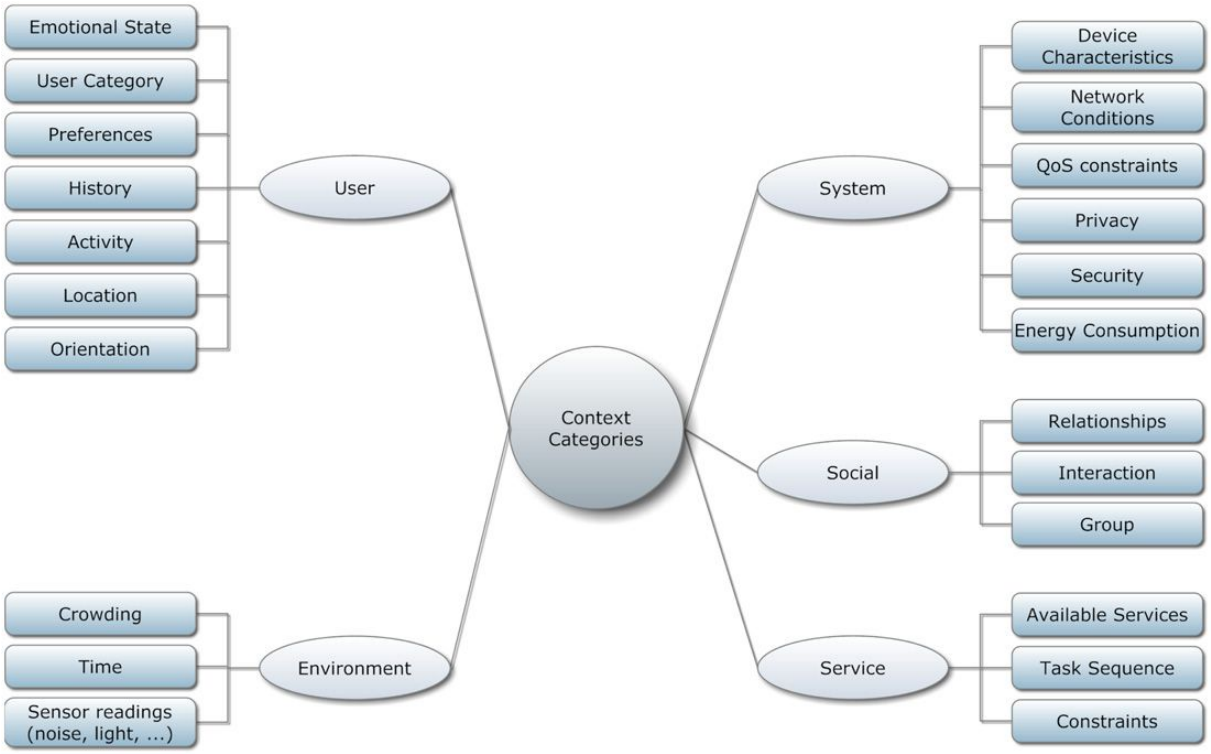

Figura 3.1: Classificação de Contexto.

Fonte: Emmanouilidis et al. [21]

tecnologia de comunicação. O termo está relacionado ao ambiente social do usuário (por exemplo, uma festa ou uma reunião) e a relação que pode ser estabelecida com outros usuários. A noção de contexto social deve levar em consideração tanto as experiências no mundo real quanto no virtual. Informações de contexto social podem ser extraídas por meio de redes sociais, sensores ou formulários. Essas informações são aspectos de contexto de alto nível relacionados com a dimensão social dos usuários, tais como o perfil do usuário, pessoas próximas, e sua atual situação social [3].

O contexto de serviço é qualquer informação usada para representar o estado de serviços disponíveis no ambiente em que o usuário se encontre e que queira utilizar. O estado de um serviço pode ser, por exemplo, o nível de congestionamento em uma determinada avenida (neste caso a avenida é o serviço) ou a quantidade de horas de espera para realizar uma compra em um supermercado (neste caso o supermercado é o serviço). O contexto de serviço pode ser utilizado também para guiar o usuário de maneira apropriada para a realização de uma sequência de tarefas de acordo com o estado dos serviços que ele deseja utilizar. Esse contexto ainda leva em consideração restrições dos serviços (por exemplo, o horário de atendimento do

serviço ou a interdependência entre serviços), as quais podem afetar a realização de uma dada tarefa.

Uma outra terminologia utilizada é o chamado Contexto Pervasivo, sendo aquele obtido a partir de sensores de hardware nos dispositivos móveis, os quais podem ser de vários tipos, como por exemplo: luminosidade, visual (câmera), áudio, movimento ou acelerômetro, localização, toque, temperatura, físicos (bio-sensores), dentre outros [5]. Dados gerados a partir desses sensores podem ser usados individualmente ou de forma combinada para inferir a situação do usuário. Essa situação pode ser, por exemplo, a (in)disponibilidade do usuário para realizar alguma atividade, seu estado de saúde ou o lugar em que se encontra (por exemplo, restaurante, residência ou local de trabalho).

Como visto, conceitos de redes sociais se unem à computação móvel e ciência de contexto e, a partir disso, Schuster et al. [53] desenvolveram o conceito de Contexto Social Pervasivo. O contexto social pervasivo de um individuo é o conjunto de informações que surgem a partir de interações diretas e indiretas entre pessoas que carregam dispositivos móveis equipados com sensores e que estejam conectadas através de uma mesma rede social. Os autores ainda classificaram e diferenciaram várias formas em que o contexto social pervasivo pode ser utilizado baseados nas *W5H Questions* [43], como visto abaixo:

- Quem - ***W**ho*: expressa quem são os participantes envolvidos no consumo e produção das informações de contexto;
- O que - ***W**hat*: diz respeito a qual tipo de contexto é utilizado ou se é importante para a aplicação;
- Onde - ***W**here*: relacionado a onde (localização física) os laços ou interações sociais são estabelecidos;
- Quando - ***W**hen*: caracterização das interações entre usuários e as informações de contexto que eles produziram em uma perspectiva temporal;
- Porquê - ***W**hy*: expressa o porquê uma informação de contexto é usada, determinando a causa ou razão dela estar sendo usada pela aplicação. Nesse caso, isso é bem relacionado ao objetivo da aplicação;

- Como - ***H**ow*: expressa como a informação de contexto (originada a partir do mundo real, mundo virtual ou de ambos) pode influenciar ou comprometer aplicações.

3.2 Modelos de Representação de Contexto

Na literatura encontram-se propostas de diferentes modelos de representação de informações de contexto. Esses modelos representam quais informações serão levadas em consideração e como elas se relacionam com a aplicação. Normalmente, modelos de contexto classificam os conceitos de um domínio ou de uma aplicação que devem ser, eventualmente, considerados como contexto (*e.g.* contexto de localização, contexto de tempo) [66]. Devido às características de dinamismo e heterogeneidade nos ambientes pervasivos, o formato no qual as informações devem ser representadas precisa contemplar algumas características, tais como [28] [27]:

- Estruturada: esta é uma característica importante no sentido de viabilizar a possibilidade de filtrar ou extrair eficientemente a informação de contexto que é relevante para a aplicação. Além disso, reduz a possibilidade de ambiguidade de atributos.

- Intercambiável: muitas vezes, as informações de contexto precisa ser trocada entre as aplicações, bem como entre os diferentes componentes da própria aplicação.

- Composta/Decomposta: compor/decompor informações de contexto é muito útil para prover manutenção de forma distribuída. Por exemplo, no caso de uma atualização da informação de contexto, ser enviada apenas parte da informação que foi modificada, evitando que seja novamente enviada toda a informação de contexto de diferentes fontes.

- Extensível: este é um conceito fundamental para a representação da informação, pois permite que a qualquer momento sejam adicionados novos parâmetros, visto que, não há um conjunto de atributos que seja identificado hoje e sirva para todas as futuras aplicações.

- Padronizada: como a informação pode vir de diferentes entidades, é fundamental que a informação seja representada de forma padronizada.

A representação de contexto e seus atributos está diretamente associada à descoberta e monitoração do contexto. É necessário disponibilizar uma representação das informações sobre seu tipo, atributos e características funcionais para que o recurso possa ser utilizado pelos clientes da infraestrutura. De acordo com Strang e Linnhoff-Popien [57] os modelos de dados relativos ao contexto podem ser agrupadas nas abordagens descritas a seguir.

3.2.1 Par Chave-Valor

A Figura 3.2 mostra um exemplo do modelo Par Chave-Valor que, como podemos perceber, é um modelo de estrutura simples. Nesta abordagem, uma informação de contexto é modelada através da utilização de uma chave como atributo de identificação e um valor associado a ela [54].

CHAVE	VALOR
DATA	01/01/2014 - 31/12/2014
HORA	00:00 - 23:59
LOCALIZAÇÃO	-2° 30.692', -44° 18.268'

AÇÃO
Exibir menssagem "Bem vindo à sua casa!"

Figura 3.2: Exemplo de contexto representado por par chave-valor

Seu processo de recuperação das informações de contexto, baseia-se em uma busca linear com a combinação exata de nomes. Apesar do seu gerenciamento simples, esse modelo não é recomendado para aplicações com estruturas complexas, pois não tem suporte a hierarquias entre os atributos. Devido a esse baixo nível de estruturação, algorítimos de recuperação de contexto não são eficazes [42].

3.2.2 Modelo de Esquema de Marcação

Diferente do Modelo anterior, a representação das informações de contexto é realizada por meio de dados hierárquicos, em quex sua especificação é textual e delimitada por *tags* de marcação. Nesse modelo, a linguagem *eXtensible Markup*

Language (XML) é usada como padrão para a modelagem dessas informações, pois facilita o compartilhamento das informações de contexto por diferentes aplicações [66]. Uma de suas desvantagens é que esse modelo não consegue solucionar ambiguidades, tornando-se assim inadequado para representar estruturas mais complexas. No trecho XML, observa-se o armazenamento do seguinte contexto: "O professor Francisco Silva está no LSD em uma sexta pela manhã".

```
1 <context>
2 <primitive>
3 <whoS><who type = "professor" value = "Francisco Silva" /> </whoS>
4 <whereS><where type = "room" value="LSD" /> </whereS>
5 <whenS><when type = "date" value = "Sexta AM" qualifier = "date" />
    </whenS>
6 </primitive>
7 </context>
```

Listagem 3.1: Representação baseado em Esquema de Marcação

3.2.3 Modelo Orientado a Objetos

Uma das grandes vantagens desse modelo é uso do encapsulamento, herança e reusabilidade, pois a estruturação das informações contextuais é feita por hierarquias de classes, como pode-se observar na Figura 3.3. Segundo [45], esse modelo não explora a descrição semântica das informações contextuais e, por isso, há uma incapacidade de implementação de algoritmos que realizem a inferência e interpretação de novos contextos a partir apenas da descrição destas informações.

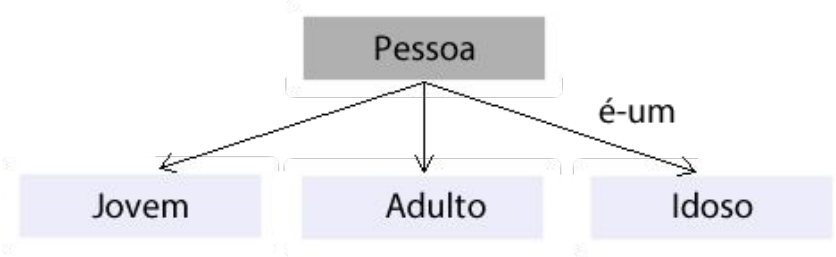

Figura 3.3: Exemplo de contexto representado por um modelo orientado a objetos.

Fonte: Soares [54].

Vieira et al. [64] afirma que as informações de contexto concentram-se no nível de interfaces, enquanto os detalhes de processamento encontram-se

encapsulados nos objetos. Outro fator interessante é que esse modelo trabalha bem com a exigência de composição distribuída. A forma de tratamento das informações contextuais (classes), as instâncias que são atualizadas ou até mesmo as novas instâncias que são atribuídas (objetos) no sistema, é distribuída, ou seja, tratados de diferentes formas.

A utilização de modelos baseados em objetos, para a representação de informações de contexto, não explora a descrição semântica das mesmas e são, portanto, incapazes de implementar algoritmos que realizem inferência e interpretação de novos contextos a partir da descrição destas informações.

3.2.4 Modelo Baseado em Lógica

Nos modelos baseados em lógica, a lógica define as condições em que uma expressão de conclusão ou fato pode ser derivada, um processo conhecido como raciocínio ou inferência, de um conjunto de outras expressões ou fatos. Para descrever essas condições em um conjunto de regras, um sistema formal é aplicado. Em um modelo de contexto baseado em lógica, o contexto, consequentemente, é definido como fatos, expressões e regras. Nesse modelo existem algumas abordagens distintas, como a Formalização de Contexto proposta por McCarthy [41], em que o contexto é introduzido como entidade matemática abstrata com propriedades úteis para a inteligência artificial.

Segundo Thomas e Linnhoff-Popien [57], modelos baseados em lógica possuem validação parcial e são difíceis de manter. Seu nível de formalidade é extremamente alto, mas sem validação parcial a especificação do conhecimento contextual dentro desse modelo é muito suscetível a erros. Os dados de contexto são adicionados, atualizados e excluídos de um sistema baseado em lógica a partir dos fatos, ou inferidos a partir de regras do sistema. Sua desvantagem é que geralmente esse modelo não oferece praticidade para validar os dados.

Existem alguns tipos de abordagem para modelagem de contexto baseado em lógica, sendo uma das primeiras pesquisas a realizada por McCarthy [41]. Em sua abordagem, McCarthy evita definir exatamente o que é contexto, e tenta prover uma formalização que viabilize o uso de axiomas simples para fenômenos de senso comum para representar contextos envolvendo poucas suposições. Outra abordagem

desse modelo é feito em [16], em que a representação de contexto é feita com o uso de predicados de primeira classe.

3.2.5 Modelo baseado em Ontologias

A ontologia é usada para representar o contexto e também para tirar proveito da capacidade de expressar relações mais complexas, como por exemplo a validação dos dados que é normalmente expressa pela imposição de restrições da ontologia [8]. Esse modelo consegue agregar conceitos e fatos. Com o uso de seus padrões é mais fácil o reuso e compartilhamento das informações de contexto. Um dos grandes interesses na construção de ontologias é ganhar conhecimento sobre o mundo real processável por máquinas [65]. Uma das primeiras abordagens desse tipo de modelo, foi proposta por [44].

Segundo [54] aplicações baseadas em ontologia permitem um nível de complexidade maior, no entanto, é necessário a estruturação de vocabulários e a associação da semântica para englobar grande parte do domínio ou até mesmo em todo seu domínio de atuação. Em [42] podemos encontrar duas principais vantagens desse modelo: i) melhora o compartilhamento de dados, eliminando as fontes de ambiguidades (o mesmo significado para um determinado conceito) e ii) possibilita criar raciocínio lógico facilmente, usando uma lógica descritiva relacionada (deduzir fatos implícitos de contexto, ou seja, modelar fatos de contexto através de experiências acumuladas pelo sistema).

3.3 Aquisição e Distribuição de Dados de Contexto

A aquisição e distribuição de dados de contexto refere-se ao processo de monitorar, capturar e/ou obter informações de diversas fontes de contexto. Essas fontes podem estar associadas ao ambiente físico, ao ambiente virtual de trabalho, a bases de dados existentes, a perfis do usuário, ou ainda ao próprio usuário, que pode informar diretamente ao sistema seu contexto atual. As fontes de contexto se encontram, em geral, distribuídas (podendo ser móveis), o que implica em maiores cuidados com aspectos espaciais e de conexão. Assim, a informação contextual pode ser adquirida a partir de diversas fontes, tais como:

1. Sensores físicos ou de hardware, que podem captar informações sobre o ambiente físico como localização, orientação e situação (por exemplo, GPS, microfones, câmeras);

2. Sensores lógicos, como agentes inteligentes ou serviços, que são capazes de monitorar e coletar informação contextual sobre o ambiente virtual de trabalho do usuário, da tarefa corrente ou do processo em andamento;

3. Entrada explícita, ou seja, algumas aplicações esperam que o usuário explicite uma informação relevante a uma dada tarefa ou, ainda, pode definir suas preferências em relação ao andamento de um processo;

4. Fontes persistentes de contexto, que mantêm informações contextuais, que não mudam frequentemente como, por exemplo: o histórico profissional do usuário, suas tarefas desenvolvidas, dados do ambiente computacional como a indicação do tipo de um dispositivo de computação ou canal de comunicação.

Os sistemas sensíveis ao contexto devem viabilizar formas de adquirir o contexto do usuário de forma mais automática possível, sem que o usuário tenha que ser questionado insistentemente sobre o contexto em que se encontra. Além disso, é essencial que os serviços de aquisição de contexto permaneçam constantemente disponíveis e é desejável que sejam implementados de forma independente das aplicações que os utilizem. Assim, diversas aplicações podem fazer uso das mesmas informações de contexto, de modo compartilhado, sem se preocupar em como aquelas informações foram adquiridas.

À medida que os ambientes tornam-se cada vez mais monitorados, por exemplo com sensores, uma maior quantidade e diversidade de informações de contexto pode ser adquirida, distribuída e compartilhada. A comunicação entre sensores e aplicações sensíveis a contexto deve ser, de modo geral, transparente. Uma vez que não haja essa transparência o desenvolvedor deve especificar e implementar um protocolo de comunicação e um esquema de codificação e decodificação para a transmissão de informações de contexto.

Aplicações sensíveis ao contexto apenas tem que ter a responsabilidade de produzir e publicar dados de contexto e declarar os seus interesses em recebê-los do *middleware*, enquanto o *middleware* assume responsabilidade de distribuição

transparente, o qual executa operações de gestão específicas para distribuir dados de contexto, tais como o encaminhamento dos dados de contexto transparente produzidos para todas as aplicações interessadas em recebê-los.

Promover a escalabilidade do sistema e a disponibilidade de dados de contexto, a produção de dados de contexto e o consumo deverá ser possível em momentos diferentes, ou seja, a comunicação deve ser assíncrona e anônima entre produtores e consumidores de contexto. Bellavista et al. [8], afirmam que a distribuição de dados de contexto é responsabilidade do *middleware* ciente de contexto, o qual torna possível a injeção dos dados de contexto no sistema e a sua entrega automática a todas as entidades que tenham manifestado qualquer forma de interesse naqueles dados de contexto. A seguir, são descritos os paradigmas de distribuição de contexto.

3.3.1 Distribuição Baseada no Modelo Publish/Subscribe

O paradigma *publish/subscribe* é uma alternativa possível para o tratamento de aplicações móveis em larga escala [23]. Eventos contêm dados que descrevem uma requisição ou mensagem e são propagados a partir dos componentes emissores, chamados de publicadores, para os componentes receptores, denominados subscritores. Os publicadores são os agentes que enviam informações a um componente central, enquanto os subscritores expressam seu interesse no recebimento de eventos particulares. O *broker* é o componente central responsável por registrar todas as subscrições, comparar as publicações com todas as subscrições e notificar os subscritores interessados. Arquiteturas baseadas em eventos *publish/subscribe* oferecem um modelo de coordenação com fraco acoplamento entre os publicadores e subscritores, em que a notificação assíncrona de eventos é naturalmente suportada.

3.3.2 Distribuição Baseado em um Espaço de Tuplas

Como discutido anteriormente, modelos de coordenação baseados em passagem de mensagens, possuem a desvantagem de acoplamento forte entre os participantes. Nesses modelos, é necessário que o emissor tenha conhecimento da identidade exata e do endereço do receptor. Além disso, existe a necessidade de sincronização, isto é, o emissor precisa esperar até que o receptor esteja pronto para

trocar de informações. Em sistemas abertos, esta característica é bastante restritiva. É necessária a utilização de um estilo de computação mais desacoplado. A computação deve prosseguir mesmo na presença de desconexão e a conectividade deve ser explorada sempre que se tornar disponível.

Uma alternativa para evitar o problema de forte acoplamento é o conceito de espaço de tuplas. Um espaço de tuplas é espaço de memória compartilhado globalmente e distribuído por todos os processos e *hosts* participantes. Os processos que usam esse modelo se comunicam gerando tuplas que são enviados ao espaço de tuplas. Tuplas são dados estruturados tipados, como por exemplo, objetos em C++ e Java, em que cada tupla é formada por uma coleção de campos de dados tipados e representa uma parte coesa da informação.

Em um sistema baseado em espaço de tuplas, todas as comunicações interprocesso são exclusivamente realizadas utilizando este espaço e qualquer processo que usa um espaço de tuplas possui a habilidade para acessar todas as tuplas que o espaço contém, inserir dinamicamente novas tuplas, encontrar associações para antituplas não destrutivas e remover tuplas gerando antituplas destrutivas associadas. Sistemas baseados em espaços de tuplas provaram sua habilidade para facilitar a comunicação em ambientes móveis. Essa forma de comunicação é bastante adequada às configurações móveis nos quais estão envolvidas a mobilidade lógica e a física [23].

3.4 Qualidade de Contexto

Os serviços de distribuição de contexto disponibilizam uma infraestrutura de suporte para coleta, gerenciamento e distribuição das informações de contexto sobre uma série de temas, que podem estar relacionados ao usuário, a objetos ou até mesmo ao ambiente. Tais serviços adquirem informações de contexto de várias fontes coletadas por terceiros, as quais geralmente fornecem os dados contextuais.

Esses dados necessitam de qualidade, principalmente quando envolvidos no âmbito da saúde. Aplicações ciente de contexto na área da saúde não podem receber dados sem um alto nível de qualidade, pois as tomadas de decisões dos profissionais da saúde baseadas nesses dados são cruciais, atingindo diretamente o estado de saúde do paciente. Por exemplo, considere um paciente com hipertensão

arterial[1] que necessita de acompanhamento em tempo real. As informações enviadas aos profissionais da saúde deverão ser as mais recentes, pois se houver atraso nas informações recebidas, haverá uma tomada de decisão equivocada, influenciando diretamente no quadro de saúde do paciente.

Outra questão relacionada à qualidade dos dados de contexto são as limitações dos consumidores para processar a quantidade de mensagens que os provedores conseguem enviar. Supomos que uma aplicação só consiga processar cada mensagem com um intervalo de três segundos, mas as mensagens às são enviadas a cada segundo. Além de causar um tráfego desnecessário na rede, a aplicação pode perder desempenho devido a quantidade de mensagens a serem processadas. Nesse contexto a qualidade é relevante, pois é necessário saber a demanda da aplicação, ou seja, o quanto a aplicação quer e pode consumir.

Qualidade de Contexto (QoC) é qualquer dado que descreve a qualidade de uma determinada informação de contexto [12]. QoC é usualmente definida em parâmetros que expressam os requisitos de qualidade e propriedades para dados de contexto [8]. Os parâmetros de QoC são utilizados para obter informações de contexto com uma estimativa correta quanto a qualidade da informação. O correto funcionamento de aplicações cientes de contexto depende, principalmente, da confiabilidade e utilidade das informações de contexto. Se os requisitos de QoC exigidos pelas aplicações não são atendidos pelo *middleware* responsável pela distribuição dos dados de contexto, isso pode comprometer o bom funcionamento das mesmas.

No trabalho de Buchholz et al. [12] são mencionadas algumas diferenças entre QoC, QoS e *Qualidade de Dispositivo* (QoD). QoC descreve a qualidade da informação de contexto, QoS indica à qualidade de um serviço. QoS é qualquer dado ou informação que descreve o quão bem um serviço é realizado. Como os serviços são executados em componentes de hardware, estes componentes podem ter certa qualidade, chamada QoD. QoD é qualquer informação sobre as características técnicas e capacidades de um dispositivo.

[1]Hipertensão Arterial são níveis pressóricos iguais ou acima de 140 mmHg (máxima) ou de 90 mmHg (mínima), devidamente avaliados pelo médico, em momentos diferentes. - http://www.intermedica.com.br/qualivida/doencas-cronicas/hipertensao-arterial

As aplicações móveis cientes de contexto que possuem enfoque em aplicações da área da saúde, necessitam de informações de contexto dotadas de alto nível de confiança, pois podem envolver circunstâncias, de certa forma, delicadas, como um sistema de acompanhamento de pacientes com problemas cardíacos. Outra grande dificuldade que merece ser destacada é a qualidade das redes móveis, pois existem várias limitações, como a largura da banda, intermitência do sinal, menor área de cobertura, dentre outros.

Saber o nível de importância de cada informação de contexto (por exemplo, uma aplicação pode preferir um nível de precisão maior para o batimento cardíaco do paciente do que a precisão da localização do mesmo) é fundamental. É preciso que se tenha conhecimento sobre a real necessidade de cada aplicação e se as informações de contexto consumidas atendem aos níveis de qualidade solicitados.

Uma nova definição de QoC é feita por [40]. Afirma o autor que o termo indica o grau de conformidade da coleta de contexto pelo sensor para a situação prevalente do ambiente e as exigências de um consumidor de contexto específico. Na figura 3.4 [40] cria um novo modelo de processamento de QoC. Tal modelo possui três diferentes camadas para o processamento das informações com QoC.

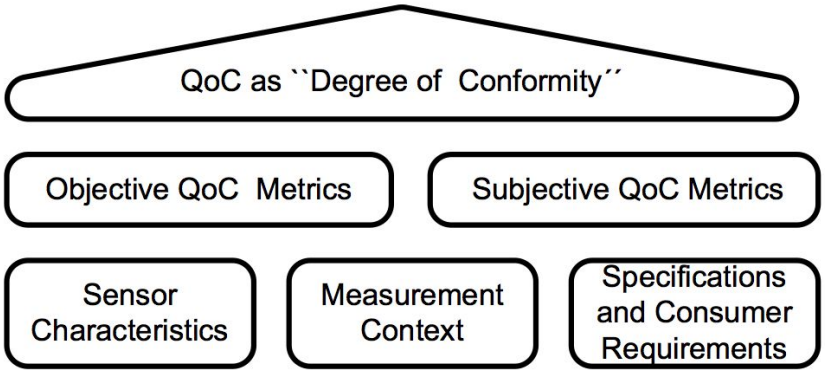

Figura 3.4: Modelo de processamento de QoC.

Fonte: Manzoor et al. [40]

A camada mais baixa é a origem da QoC, formada pelos dados utilizados pela camada superior. Compõe-se de três sub-categorias, a saber: i) características do sensor - são informações sobre o sensor que podem afetar a qualidade da informação de contexto fornecido pelo dispositivo, como a acurácia, precisão, granularidade, período de tempo, estado do sensor e o alcance dele; ii) medição do contexto - mostra as informações relacionadas a uma medição específica, a exemplo do tempo de medição,

do sensor de localização, das informações de localização de uma entidade, em suma, referentes aos atributos específicos do contexto de um objeto; iii) especificações e exigências do consumidor - o consumidor faz um detalhamento de suas exigências sobre a qualidade das informações de contexto, como o tempo de validade da informação, atributos necessários, valor crítico e nível de acesso.

A camada superior é representada por dois tipos de métricas: as métricas de QoC objetivas e as de QoC subjetivas. As primeiras demonstram a qualidade de contexto como uma quantidade independente e seu cálculo envolve características do sensor e medição do contexto; as subjetivas, a qualidade do contexto para utilização de um consumidor específico para uma determinada finalidade.

No âmbito da computação ubíqua, informações de contexto são coletadas de várias fontes, (*e.g.* podem ser fornecidas por usuários e obtidas de sensores). Os parâmetros de QoC são associados a informações de contexto e têm o propósito de identificar a qualidade da informação. Buchholz et al [12], enumeram alguns parâmetros de QoC , (*e.g.* atualidade, frequência, precisão, taxa de atualização, etc), que são descritos a seguir.

- **Acurácia** (*Accuracy*): é o grau de exatidão do contexto ou a capacidade do sensor em medir a quantidade aproximada ao valor real. [69] assinala que *inaccuracy*, ou erro absoluto de um sensor físico, pode ser calculado pela diferença entre o valor real e o valor mensurado pelo sensor. Temos, então, a seguinte equação,

$$[E = T - M] \tag{3.1}$$

  na qual, E é o erro na medição, T é o valor real e M é o valor mensurado pelo sensor. Geralmente, o valor real é acordado antes da medição ou em alguns casos é uma verdade absoluta. Já a Acurácia de um sensor físico, tem seu valor calculado através da equação,

$$[Accuracy = 1 - \frac{|E|}{T}] \tag{3.2}$$

  sendo E o valor da inacurácia e T, o valor real. Entende-se que a acurácia é calculada subtraindo o erro relativo de 1. Outra acurácia importante nesta definição é a do sensor virtual, o qual pode ser calculado com a seguinte equação

$$[Accuracy = \frac{Número\ de\ instâncias\ classificadas\ como\ corretas}{Número\ total\ de\ instâncias}] \tag{3.3}$$

- **Atualidade** (*Freshness*): indica a idade da informação recebida, ou seja, o tempo decorrido desde que a informação foi coletada até a sua entrega ao consumidor;
- **Frequência** (*Frequency*): indica a quantidade de atualizações que a aplicação deseja receber a cada intervalo de tempo;
- **Precisão** (*Precision*): é o grau de exatidão de uma medição. Isto indica a capacidade de um sensor para dar a mesma leitura que a mesma quantidade de medição sob as mesmas condições;
- **Taxa de atualização** (*Refresh Rate*): indica quantidade de atualizações recebidas que aplicação deseja processar a cada intervalo, independente da frequência com a qual essas atualizações são produzidas.

3.5 Considerações Finais

Este capítulo apresentou os conceitos relacionados à computação ciente de contexto, enumerando pontos relevantes e inerentes à construção de sistemas cientes de contexto. O objetivo foi prover uma base para facilitar a compreensão de alguns conceitos sobre ciência de contexto e sua aplicação à computação ubíqua.

Para tanto, foi discutida a relevância do estudo sobre contexto encontrado na literatura, foram apresentadas ainda definições sobre contexto e sobre QoC. Foram descritos também requisitos de contexto que devem ser considerados para o desenvolvimento de aplicações cientes de contexto.

Um requisito fundamental que se deve observar para a construção de um sistema ciente de contexto e, principalmente, para viabilizar o desenvolvimento de uma infraestrutura de *middleware* que controle o gerenciamento de informações de contexto, é a existência de um modelo formal de representação que permita a interoperabilidade e o compartilhamento de informações entre diferentes aplicações cientes de contexto, podendo considerar também parâmetros de QoC na definição desses requisitos de contexto.

4 ARQUITETURA CENTRADO EM DADOS

Na arquitetura centrada nos dados o intercâmbio de informações entre aplicações se dá por meio de um espaço de dados. O espaço de dados é uma abstração composta por um conjunto de dados publicados e subscritos, em que a geração de informação consiste em publicar (escrever) no espaço de dados enquanto que o consumo da informação (subscrição) se realiza lendo dados deste espaço [62]. Cada informação distribuída no espaço de dados é previamente estabelecida através de um modelo de dados. Nesta abordagem, controla-se diretamente os dados que são distribuídos na rede, diferente de outras abordagens em que é necessária a configuração de clientes e servidores ou a criação de objetos para acessar dados remotamente. Assim, os desenvolvedores não escrevem códigos específicos para invocar métodos, por exemplo, mas constroem uma espécie de "dicionário de dados", definindo quem precisará de quais dados [56]. No MobileHealthNet estes dados são distribuídos em forma de tópicos, possuindo um nome, usado para identificação do dado, e um conteúdo (e.g, tipos básicos da linguagem Java).

Esse paradigma cria a ilusão de um espaço global de dados, no qual estão contidos os diversos dados acessados pelas aplicações por meio de simples operações de leitura e escrita. Isso traz uma grande vantagem para a arquitetura centrada em dados: o baixo acoplamento, pois publicadores não precisam saber da existência dos subscritores de suas informações, ou seja, o desacoplamento é de ordem referencial, temporal e espacial. Caso se adote uma arquitetura centrada em dados e completamente distribuída, a implementação desse modelo pode trazer escalabilidade, desde que permita um grande número de consumidores e publicadores compartilhando informações entre si. Este capítulo apresenta uma introdução sobre modelo *Data-Centric Publish Subscribe* (DCPS), abordando o *middleware* DDS, suas principais propriedades e arquitetura. Neste capítulo também são detalhadas as políticas de QoS definidas pelo DDS.

4.1 Modelo *Publish/Subscribe* Centrado em Dados

Bellavista et al. [8] sugerem que o *middleware* ciente de contexto deve ter uma arquitetura centrada em dados, que é uma alternativa possível para o tratamento de aplicações móveis em larga escala [23]. A distribuição é baseada em eventos que contêm dados que descrevem uma requisição de interesse por dados de contexto e são propagados a partir dos componentes emissores (produtores), denominadores publicadores, para os componentes receptores (consumidores), denominados subscritores. Os publicadores (*publishers*) são usualmente sensores que enviam informação para o *middleware* ciente de contexto, enquanto os subscritores (*subscribers*) expressam seu interesse no recebimento de eventos particulares.

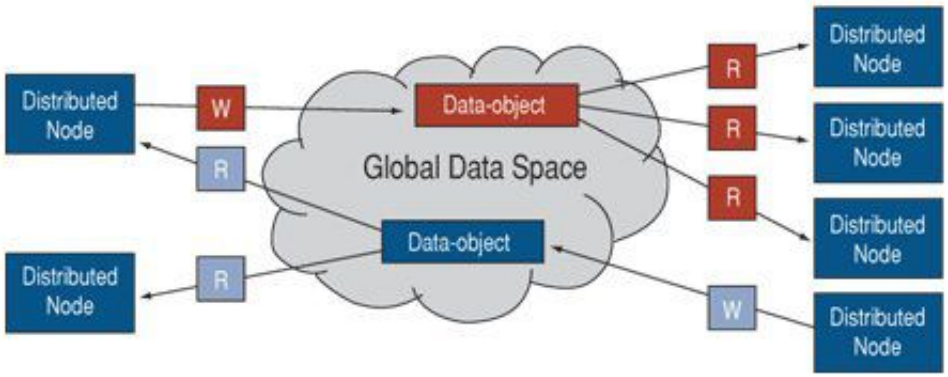

Figura 4.1: Espaço Global de Dados.

Fonte: Schneider [56].

O modelo DCPS baseia-se no paradigma *publish/subscribe* de troca de mensagens centrado em dados, no qual as informações são representadas por uma estrutura de dados. Como nesta abordagem existe um espaço de dados para a troca de informações na rede, no qual estes dados são basicamente publicados ou escritos e subscritos ou lidos, é natural o uso do paradigma *publish/subscribe* para interação entre consumidores e publicadores. Conforme ilustrado na Figura 4.1, esse paradigma cria a ilusão de um espaço global de dados, no qual estão contidos os diversos dados acessados pelas aplicações utilizando simples operações de leitura e escrita.

Esse modelo permite que cada dado da aplicação esteja associado com um grupo de subscritores. Portanto, quando um determinado subscritor registra interesse a um determinado dado, ele estará se associando a um grupo que está relacionado com

o respectivo dado. Uma característica dessa estratégia é que um mesmo nó pode ser publicador e subscritor para vários dados. O uso do modelo centrado nos dados foi escolhido para comunicação entre os componentes de *software* no MobileHealthNet.

Os dados são o centro de qualquer *middleware* de comunicação e é especialmente importante para aqueles que utilizam o modelo DCPS. O modelo DCPS simplifica a complexa programação para ambientes distribuídos. Especificamente, é a infraestrutura de comunicação baseado em DCPS que gerencia automaticamente a entrega das mensagens sem requerer qualquer intervenção das aplicações. Esse gerenciamento inclui quais nós devem receber uma mensagem, a localização de quem envia ou recebe mensagens e o tratamento de falhas na comunicação. Ao utilizar um modelo de comunicação DCPS, o desenvolvedor se preocupa somente com os tipos de dados que serão compartilhados entre as aplicações.

4.2 Data Distribution Service

No núcleo da infraestrutura de comunicação do MobileHealthNet , optou-se por utilizar a especificação DDS[1] [13] [51] [67] [30]. O DDS é uma especificação *publish/subscribe* que visa permitir alta escalabilidade, alto desempenho e troca de dados com interoperabilidade entre publicadores e subscritores, e foi idealizado sobre um modelo DCPS baseado em tópicos.

O DDS possui sua arquitetura completamente distribuída. Aplicações interagem entre si por de mecanismos de interação *publish/subscribe* e podem compartilhar dados entre si sem a intervenção de servidores, compartilhando informações de forma ponto-a-ponto. Foi projetado para [71]:

- Prover independência de local via protocolos anônimos do tipo *publish/subscribe* que permitem a comunicação entre publicadores e subscritores remotos;
- Escalabilidade, suportando um grande número de dados, leitores e escritores de dados;
- Portabilidade e interoperabilidade entre plataformas.

[1]Data Distribution Service for Real-Time Systems Specification em:<http://www.omg.org/spec/DDS/1.2/PDF/>. acesso em janeiro de 2014

A especificação DDS foi formalmente adotada pela OMG em 2004 e em pouco tempo se tornou referência em tecnologia *publish/subscribe* para distribuição de grandes volumes de dados, de modo confiável com baixas latências em aplicações como processadores de radar, drones de voo e terra, sistemas de gerenciamento de combate, controle e gerenciamento de tráfico aéreo, etc.

Essa especificação possui menos camadas que um SOA tradicional [71], como CORBA, .NET e J2EE, o que reduz significativamente a latência de comunicação (o tempo entre a publicação e o recebimento de uma resposta pelo *subscriber*). Além disso, a independência de local, alta escalabilidade, portabilidade, interoperabilidade, alto desempenho e as políticas de qualidade de serviço são grandes atrativos que a especificação DDS fornece. Dentre algumas implementações existentes, o CoreDx[2] foi escolhido como o *middleware* DDS DCPS, pois foi concebido para dispositivos embarcados, utilizando poucos recursos computacionais.

4.2.1 Entidades DDS

O DDS possui as seguintes entidades envolvidas na troca de informações:

- *Domains*: Aplicações DDS enviam e recebem dados dentro de um domínio. Somente participantes do mesmo domínio podem se comunicar, o que ajuda a isolar e otimizar a comunicação dentro do espaço global de dados.
- *Data Writers*: Aplicações podem utilizar *data writers* para publicar dados no espaço global de dados em um determinado domínio.
- *Data Readers*: Recebem os dados publicados pelos *data writers*.
- *Publisher*: Cria e gerencia um grupo de *data writers*.
- *Subscriber*: Cria e gerencia um grupo de *data readers*.
- *Topics*: Responsável pela ligação entre um *data writer* a um *data reader*. A comunicação ocorre somente se o tópico publicado por um *data writer* equivale ao tópico subscrito pelo *data reader*. A comunicação via tópico é anônima e

[2]Twinoaks Computing - CoreDX. Disponivel em:<http://www.twinoakscomputing.com/coredx>. acesso em janeiro de 2014

transparente, *publishers* e *subscribers* não precisam se preocupar como os tópicos são criados nem com quem está escrevendo ou lendo o tópico, pois o *middleware* DDS DCPS gerencia estes problemas.

Essas entidades são essenciais para o desenvolvimento de aplicações utilizando a especificação DDS. A *DomainParticipantFactory* é a fábrica de domínios (*DomainParticipant*), ou seja, esta entidade é a única em todo ambiente DDS capaz de criar domínios. O participante de domínio (*DomainPartipant*) é a fábrica de Publicadores (*Publishers*), Subscritores (*Subscribers*) e Tópicos (*Topics*). Um único domínio pode criar diversos tipos destas entidades. Publicadores são fábricas de Escritores de Dados (*Data Writers*) e Subscritores são fábricas de Leitores de Dados (*Data Readers*). Os tópicos (*Topics*) são as entidades que ligam os escritores aos leitores de dados.

A interação entre entidades DDS pode ser melhor visualizada pela Figura 4.2, a qual descreve um único domínio no qual ocorre a troca de informações entre as entidades. Cada entidade pode possuir políticas de QoS específicas e contém mecanismos para armazenamento dos dados lançados no espaço global de dados (*Data Store + QoS Mechanisms*). Na camada mais inferior está o protocolo de transporte, nativamente UDP, que permite o envio de mensagens de maneira *unicast*, *multicast* e *broadcast*. Finalmente, na camada física, tem-se a rede de comunicação propriamente dita.

4.2.2 Política de Qualidade de Serviço

A especificação DDS aborda um rico conjunto de políticas de QoS com base em contrato entre *Publisher* e *Subscriber*, incluindo, entre outros: durabilidade, tempo de vida, prazo e prioridade de transporte. Políticas de QoS fornecem um mecanismo para as aplicações distribuídas poderem controlar o comportamento de uma entidade [36].

As entidades da especificação DDS incluem tópicos, que descrevem o tipo de dados a ser escrito ou lido; Os *DataReaders*, que registram o interesse em receber os dados de um tópico específico; e *DataWriters*, que publica os dados ou instâncias para tópicos específicos. Várias propriedades dessas entidades podem ser configuradas

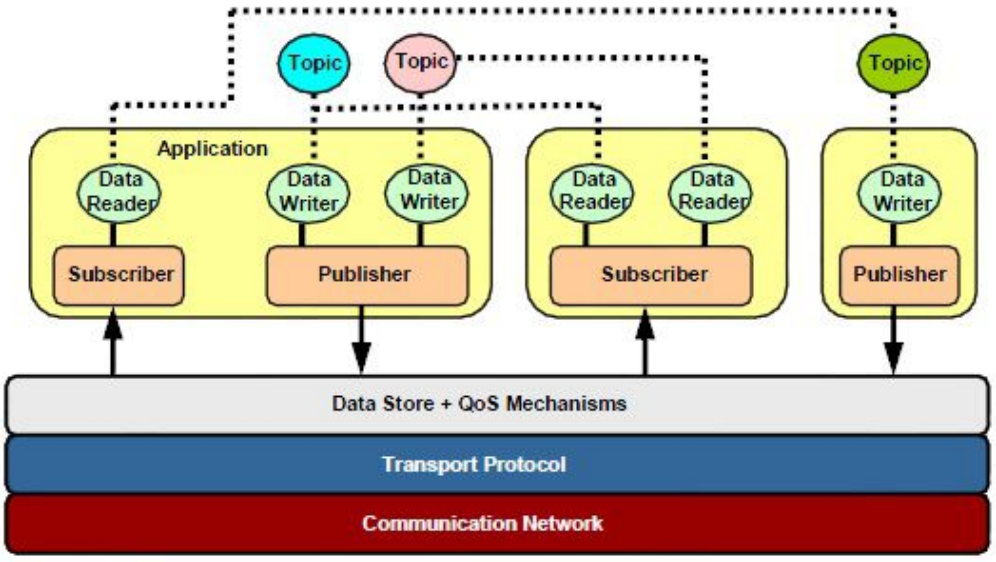

Figura 4.2: Arquitetura DDS.

Fonte: Xiong et al. [71].

usando combinações das 22 Políticas de QoS presente na especificação DDS. Em vários casos, para que a comunicação ocorra de forma eficiente, a política de QoS do lado do *DataWriter* deve ser compatível com a política correspondente do lado do *DataReader*.

Por disponibilizar um conjunto de políticas de QoS, o DDS proporciona às aplicações clientes a capacidade de controlar e limitar a utilização de recursos, tais como, consumo da largura de banda e memória, e muitas propriedades não funcionais, tais como, persistência, confiança e pontualidade. E nesse trabalho, destacamos as seguintes políticas de QoS:

1. *RESOURCE_LIMITS*: permite controlar a quantidade de mensagens armazenadas na cache das entidades;

2. *TIME_BASED_FILTER*: permite aos *DataReaders* limitar o número de dados recebidos dentro de um intervalo de tempo;

3. *DEADLINE*: garante que publicador publicará pelo menos uma atualização no tópico dentro de um intervalo de tempo;

4. *LATENCY_BUDGET*: especifica um atraso aceitável entre o tempo no qual o *DataWriter* grava um dado e o tempo no qual o *DataReader* recebe a notificação de que esse dado está disponível;

5. *DURABILITY*: permite às entidades determinar se os tópicos serão salvos pelo *middleware* DDS;

6. *LIFESPAN*: permite definir o prazo de validade para os dados publicados;

7. *HISTORY*: especifica a quantidade de dados que serão armazenados para uma entrega posterior;

8. *RELIABILITY* permite à aplicação controlar o nível de confiabilidade na entrega dos dados;

9. *DESTINATION_ORDER*: permite controlar a ordem de chegada dos dados publicados;

10. *OWNERSHIP*: permite controlar o número de publicadores com permissão para um determinado tópico.

4.3 Considerações Finais

O modelo DCPS fornece as funcionalidades necessárias para o desenvolvimento de aplicações cientes de contexto, garantindo um fraco acoplamento entre os consumidores e produtores de dados. Em uma infraestrutura *publish/subscribe*, os *publishers* produzem as publicações, unidades de informação não endereçadas que são entregues ao sistema. Os *subscribers*, por sua vez, expressam o seu interesse em publicações com determinadas características. Cabe à infraestrutura *publish/subscribe* a responsabilidade de entregar as publicações aos *subscribers* que nelas estejam interessados.

Na definição do *middleware* proposto neste trabalho de mestrado, adotou-se uma abordagem orientada a dados para a definição do modelo de comunicação, no qual os dados são transmitidos na forma de tópicos, formando o dicionário de dados do MobileHealthNet. Cada tópico possui informações de entidades ou de ações realizadas por estas e permitem a troca de dados entre clientes e provedores de serviços.

O modelo orientado a dados molda-se perfeitamente às necessidades do projeto MobileHealthNet , pois a notificação da produção de informação de interesse

contribui para a redução do número de mensagens desnecessárias. Essa característica é útil para aplicações móveis, cuja comunicação está sujeita à conectividade intermitente e eventuais desconexões. Além disso, o DDS suporta diversas políticas de QoS relacionadas à distribuição de dados. Essas políticas são utilizadas em nossa proposta para atender alguns parâmetros de QoC.

5 MOBILEHEALTHNET CONTEXT SERVICE

Este capítulo apresenta o *middleware* MobileHealthNet Context Service - MHNCS, cuja principal responsabilidade é a coleta e distribuição de informações de contexto para aplicações ubíquas. O MHNCS disponibiliza uma linguagem flexível que dá suporte à especificação de requisitos de contexto com qualidade (QoC), bem como a obtenção, validação, e distribuição de dados de contexto, considerando parâmetros de QoC, esses parâmetros foram descritos no capitulo 3, especificados pelo desenvolvedor da aplicação ubíqua que faz uso do *middleware*. O objetivo principal deste trabalho é criar um *middleware* que gerencie a distribuição de contexto e facilite o desenvolvimento de aplicações móveis cientes de contexto contemplando requisitos específicos da área da saúde.

A infraestrutura proposta neste trabalho está inserida na arquitetura do MobileHealthNet descrito no capítulo 2, o qual corresponde ao *Context Service* da camada *Core Services*, conforme a Figura 2.4. Apesar de na arquitetura do MobileHealthNet ele ser referenciado como serviço, ele pode ser instanciado e utilizado de forma independente do MobileHealthNet, o que justificaria chamá-lo de *middleware*.

5.1 Requisitos do MHNCS

O acompanhamento do estado de saúde dos pacientes através de recursos da computação ubíqua permite aos profissionais da área da saúde obter dados diariamente de seus pacientes. Em outras palavras, o suporte da mobilidade promovida pela tecnologia móvel na medicina permite o acompanhamento e o tratamento contínuo ou pontual dos pacientes, mesmo à distância, fora do ambiente hospitalar. O acompanhamento de pacientes utilizando recursos ubíquos requer importantes requisitos relacionados ao desenvolvimento de aplicações cientes de contexto [6]. O MHNCS possui os seguintes requisitos funcionais:

(i) Aquisição de dados provenientes de sensores físicos, virtuais e lógicos;

(ii) Comunicação entre os sensores e outros componentes da infraestrutura que estão fisicamente dispostos de maneira distribuída;

(iii) Controle do fluxo de entrega das informações de contexto, utilizando parâmetros de QoC, possibilitando uma economia de processamento e de energia e, consequente, o aumento do tempo de vida da bateria dos dispositivos móveis;

(iv) Distribuição das informações de contexto;

(v) Processamento ou obtenção dos dados brutos obtidos dos sensores para a transformação em informações de contexto;

(vi) Notificação de informações de contexto através de eventos.

Segundo Barua et.al. [6], a taxa de atualização das informações é um importante requisito na distribuição dos dados. Dependendo das configurações de cada sensor, e diferentes taxas de atualização poderão ser adotados. Nossa proposta permite às aplicações ubíquas definir a taxa de atualização e a frequência de recebimento dos dados.

A interoperabilidade é outro requisito importante, uma vez que a infraestrutura ciente de contexto deve permitir que as aplicações recebam facilmente as informações de contexto em um ambiente ubíquo. Em nossa proposta, a interoperabilidade é obtida pela especificação de comunicação DDS. Essa especificação garante um canal de comunicação com baixo acoplamento entre as aplicações e as fontes de contexto. Com isso, garantimos a distribuição e o compartilhamento das informações de forma transparente entre as aplicações ubíquas.

5.2 Linguagem para Especificação de Requisitos de Contexto

Em nosso trabalho propomos uma linguagem de definição de requisitos de contexto que garante flexibilidade na especificação dos mesmos. Nesta linguagem, denominada *MHN Context Language*, define quais informações de contexto são requeridas pela aplicação e, de forma opcional, parâmetros de QoC associados às

mesmas. Trata-se de um modelo descritivo de meta-informação para a representação de requisitos de contexto obtidos dos provedores de contexto.

Essa meta-informação suporta a especificação de informações relativas à descoberta e inicialização dos provedores de contexto, de modo a criar um meio universal de interoperabilidade com os provedores de contexto distribuídos em um ambiente ubíquo. Toda estrutura da linguagem foi implementada usando a especificação XML, uma linguagem neutra com relação à implementação e que permite a criação de outras linguagens a partir dela. No apêndice A é exposto o esquema *XML Schema Definition* (XSD)[1] que detalha a estrutura da *MHN Context Language* proposta. De acordo com as informações definidas na especificação, a infraestrutura do MHNCS irá localizar os provedores de contexto que melhor atendem as necessidades de cada aplicação.

A listagem 5.1 apresenta um exemplo de especificação de requisitos de contexto sem a definição de parâmetros de QoC. Neste exemplo, temos o elemento `contextInformation`, presente na linha 3, utilizado para descrever a informação de contexto. O tipo da informação é definido através do elemento `information`, o qual é composto por dois atributos: *id:* especifica a informação de contexto que a aplicação deseja receber; e o `init`: utilizado para determinar a quantidade de provedores de contexto que serão inicializados. O `init` recebe os seguintes valores: (i) *SINGLE:* define que será inicializado apenas um provedor de contexto; e (ii) *ALL:* quando a aplicação deseja inicializar todos os provedores de contexto que fornecem a mesma informação.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <requirement>
3   <contextInformation>
4     <information id="location" init="SINGLE"/>
5     <information id="time" init="ALL"/>
6    </contextInformation>
7    <qualityOfContext/>
8 </requirement>
```

Listagem 5.1: Descrição de requisitos de contexto sem QoC

[1] http://www.w3schools.com/schema/

Algumas aplicações sensíveis ao contexto tem a necessidade em obter dados do mesmo tipo de informação de sensores diferentes. Estes sensores estão dispersos sobre uma região, sendo assim, apesar de coletarem o mesmo tipo de informação os valores retornados por eles são diferentes. Sistemas sensíveis ao contexto podem realizar analises estatísticas ou inferir um novo fato através dos dados coletados. Pode-se exemplificar isto com um sistema capaz de coletar dados de diversos sensores de temperatura espalhando em uma região metropolitana e determinar a temperatura média. Dessa forma, é possível expressar várias informações de contexto e inicializar um ou vários provedores de contexto, conforme apresentado nas linhas 4 e 5.

5.2.1 Especificação de Parâmetros de QoC

Para especificar os parâmetros de QoC na *MHN Context Language* é utilizado o elemento `qualityOfContext`. O elemento *`property`* serve para descrever as propriedades relacionadas aos parâmetros de QoC suportados pela linguagem. O *`property`* possui um conjunto de atributos que serve para especificar as propriedades dos parâmetros de QoC, conforme a estrutura definida no apêndice A. A listagem 5.2 apresenta um exemplo de definição de requisitos de contexto atribuindo parâmetros de QoC. Neste exemplo, temos nas linhas 4-5 a definição das informações de contexto. Os parâmetros de QoC são definidos pelo elemento `property` linha 7-8. Na linha 7 a aplicação define que deseja receber as informações com uma taxa de atualização (**REFRESH_RATE**) de 120 segundos e, neste caso, serão entregues sempre as cinco últimas atualizações. Finalmente, na linha 8 é determinado que cada informação recebida não poderá ter uma idade igual ou superior a 120 segundos.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <requirement>
3    <contextInformation>
4      <information id="location" init="SINGLE"/>
5      <qualityOfContext>
6        <property name="REFRESH_RATE" time="120" kind="GET_LAST" amount="5"/>
7        <property name="FRESHNESS" age="120" op="lte"/>
8      </qualityOfContext>
9    </contextInformation>
10 </requirement>
```

Listagem 5.2: Descrição de requisitos de contexto com qualidade de contexto

Especificando o parâmetro de QoC Frequência

Para requerer o parâmetro de QoC frequência a aplicação utilizará atributo `name` para especificar esse parâmetro, e também terá que determinar o intervalo de tempo que deseja receber uma nova informação pelo atributo `time`. O atributo `kind` permite a aplicação definir a variante da frequência. Essa variante pode ser:

1. *MINIMUM* define que a aplicação, a cada intervalo de tempo, deseja receber pelo menos a última atualização;
2. *EXACT* define que a aplicação, a cada intervalo de tempo, deseja receber exatamente a última atualização;
3. *MAXIMUM* define que a aplicação, a cada intervalo de tempo, deseja receber pelo menos uma atualização e no máximo as N últimas atualizações. E o atributo `amount` será utilizado pela aplicação para determinar a quantidade de atualizações que deseja receber.

Especificando o parâmetro de QoC Taxa de Atualização

Para especificar o parâmetro *Refresh rate*, além de especificar o intervalo de tempo que deseja receber uma nova informação pelo atributo `time`, a aplicação também definirá a variante desse parâmetro, semelhante ao *Frequency*. E neste caso o *Refresh rate* possui dois tipo de variante:

1. *Get_All*: define que a aplicação, a cada intervalo de tempo, deseja processar todas as atualizações recebidas;
2. *Get_Last*: Define que a aplicação, a cada intervalo de tempo, deseja processar apenas as N últimas atualizações recebidas. Semelhante a *Frequency MAXIMUM*, aplicação também utiliza atributo `amount` para determinar o número de atualizações.

Especificando o parâmetro de QoC Atualidade

Com relação ao parâmetro *Freshness*, além de utilizar o atributo `name`, há dois atributos que devem ser utilizados pela aplicação que são o atributo `age` e o

atributo `op`. O atributo `age` define a idade da informação em segundos; e o atributo `op` utilizado para indicar o tipo de comparação matemática.

Especificando o parâmetro de QoC Acurácia

Para especificar o parâmetro *Accuracy* a aplicação deverá determinar grau de exatidão da informação requerida pelo atributo *variation* em conjunto com o atributo `op`.

5.3 Arquitetura do MobileHealthNet Context Service

Neste trabalho é proposto um *middleware* para o desenvolvimento de aplicações ubíquas ciente de contexto denominado MHNCS. Na proposta são utilizadas as técnicas de desenvolvimento de software orientado a serviços e composição dinâmica de componente de *software* que irão receber dados de sensores embarcados no dispositivo móvel ou sensores externos, a fim de prover informações cientes ao contexto em um ambiente ubíquo. Tal proposta tem como objetivo reduzir o grau de acoplamento entre as aplicações, bem como permitir a aquisição e distribuição das informações de contexto.

O MHNCS é uma infraestrutura de *middleware* centrado em dados com o suporte à distribuição das informações de contexto. Esta proposta está inserida no escopo do projeto MobileHealthNet. O *middleware* explora as potencialidades da computação sensível ao contexto para prover mecanismos de distribuição e notificação para as aplicações cientes de contexto, visando a integração de dispositivos móveis no acompanhamento de pacientes com doenças crônicas. O MHNCS tem como objetivos:

- Definir de uma arquitetura com o suporte à computação ciente de contexto;
- Explorar o uso de contexto na execução de aplicações ciente de contexto;
- Usar QoC para obter informações de contexto mais confiáveis para as aplicações.

A dinamicidade das aplicações cientes de contexto e a crescente integração destas aplicações nas tarefas cotidianas das pessoas geram mudanças rápidas e

constantes nos requisitos das infraestruturas de suporte a estas aplicações. Além disso, desenvolver aplicações cientes de contexto impacta em um alto custo e um grande consumo de tempo, devido à existência de uma grande variedade de informações de contexto que podem ser utilizadas nessas aplicações. Algumas dessas informações não são explícitas ou não podem ser obtidas a partir da leitura de um sensor, mas podem ser inferidas a partir de outras. Estes problemas tornam bastante complexo o desenvolvimento dessas aplicações. Sendo assim, o MHNCS proposto neste trabalho possui como objetivo solucionar alguns destes problemas.

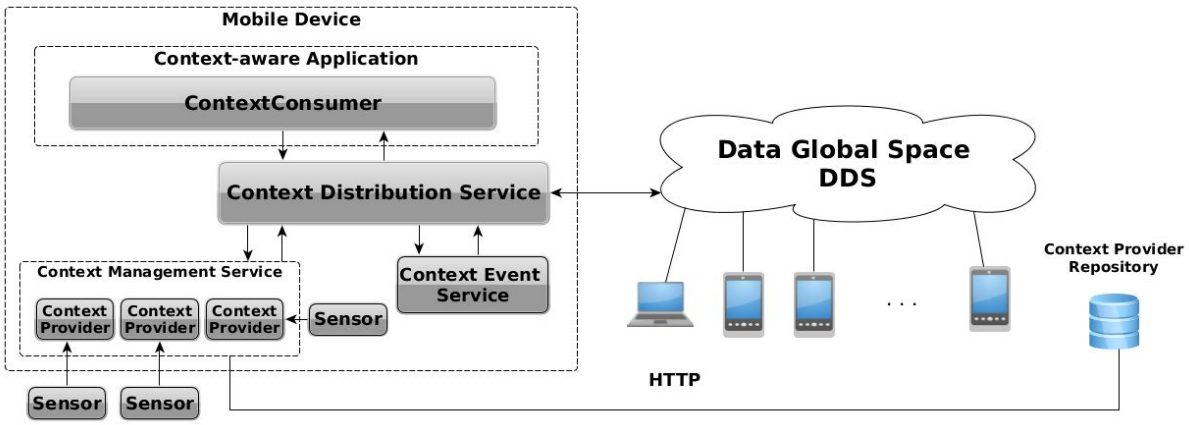

Figura 5.1: Arquitetura do MHNCS.

O MHNCS é composto por um conjunto de componentes necessários para auxiliar a integração do dispositivo com o ambiente ubíquo e disponibilizar os serviços necessários que compõem a arquitetura. Essa arquitetura, conforme a Figura 5.1, é formada por 3 componentes básicos: Módulo de Distribuição, *Context Distribution Service* (CDS), que gerencia a distribuição de contexto para os dispositivos móveis em um ambiente pervasivo; Gerenciador de Eventos, *Context Event Service* (CES), que fornece o mecanismo que avalia as informações de contexto e notifica as aplicações somente quando as condições definidas forem satisfeitas; e parte da arquitetura proposta é formada por componentes modificados, oriundos de um *middleware* voltado para o gerenciamento de provedores de contexto denominado *Context Management Service Context Management Service* (CMS) [39].

5.3.1 Context Management Service

O CMS é responsável pela comunicação entre a infraestrutura e os provedores de contexto (representados pelos *context providers* na Figura 5.1). No MHNCS, a obtenção dos dados de contexto é de responsabilidade do componente CMS. O CMS recebe as especificações e então inicia a busca por *context providers* compatíveis com os requisitos. Se não houver nenhum provedor compatível instanciado, o CMS se encarrega de instanciá-los. Para obter o dado de contexto, é necessário que o provedor se registre junto ao software embutido no sensor. Cada provedor de contexto pode obter dados a partir de um único sensor. Entretanto, o CMS pode gerenciar vários provedores de contexto e combinar os dados enviados por eles a fim de obter informações de mais alto nível.

O CMS utiliza uma arquitetura baseada em componentes, o que permite que novos provedores de contexto possam ser encontrados e carregados no serviço a partir de um repositório remoto. Dessa forma, se uma aplicação requisita alguma informação de contexto não suportada por nenhum provedor localmente disponível no dispositivo, o CMS pode fazer o download e carregar dinamicamente o componente capaz de prover a informação requisitada.

O componente responsável por atualizar as informações de contexto para as aplicações são os *context providers*. Estes enviam as mudanças ocorridas no ambiente monitorado. Dessa forma, os provedores possuem a tarefa de coletar as informações provindas de sensores (*e.g.* GPS, pressão, temperatura corporal) ou outras informações externas (*e.g.* previsão do tempo, calendário, agenda) utilizando um modelo abstrato para representar os dados de contexto.

Apoiando-se no estudo comparativo entre as abordagens para modelagem de contexto apresentado na Capítulo 3, foi utilizado um modelo abstrato baseada em Orientação a Objetos para a modelagem de contexto no MHNCS, sendo o modelo utilizado descrito no trabalho de [39]. Para a representação do contexto e das entidades que a possuem, utilizamos a classe `ContextInformationObject`. Essa classe é utilizada pelo provedor de contexto para notificar o serviço de distribuição de contexto. O Provedor de contexto notificará o CMS, que fará a publicação via serviço de distribuição de contexto do MHNCS para que a informação chegue até as aplicações ciente de contexto. O modelo de contexto adotado em nosso trabalho não

inclui informações semânticas das informações representados, mas apenas a estrutura dos mesmos, ficando a cargo das aplicações o conhecimento de como tratar cada propriedade das informações.

5.3.2 Context Distribution Service

O CDS é considerado o principal componente do *middleware* MHNCS. Nele, toda a lógica necessária para a distribuição de contexto é definida. Na distribuição dos dados de contexto entre as aplicações utilizamos especificação *publish/subscribe* DDS. Nesse modelo, cada dispositivo móvel corresponde a um nó e é capaz de receber dados de contexto destinados às aplicações que nele executam, bem como encaminhar dados destinados a aplicações em execução em outros nós do ambiente distribuído. O gerenciamento das transmissões a partir dos dispositivos móveis é realizado por meio do CDS que executa nos dispositivos como um serviço Android [2] desenvolvido para este fim. Toda a comunicação entre os nós da rede ocorre mediante autenticação prévia das partes comunicantes, a qual se dá por meio da camada *Security Services* [24], a fim de garantir a privacidade dos dados transmitidos.

O CDS é responsável por realizar a comunicação entre os consumidores e o CMS. Ele fornece um mecanismo de comunicação centrado em dados possibilitando o acesso aos dados de contexto coletados em um ambiente distribuído DDS. No DDS os dados são transmitidos em formato de tópicos, em que o dado é publicado no domínio DDS em um determinado tópico e os interessados neste dado se subscrevem a este tópico, recebendo notificações sempre que um novo dado estiver disponível. Dessa forma, é possível definir um tópico para representar cada informação de contexto monitorada pelos provedores de contexto, de forma que as aplicações irão receber dados dos tópicos aos quais se subscreveram. Essa forma de comunicação reduz o número de entregas de informações e diminui o consumo de largura de banda da rede.

Para obter os dados de contexto que foram especificados, é necessário antes solicitar autorização de acesso ao usuário que detém a privacidade e o controle das informações. A publicação de uma solicitação contém o UID do usuário solicitante (local), o UID do usuário solicitado (local ou remoto) e a especificação dos requisitos de contexto, se o usuário solicitado confirmar que deseja compartilhar as informações especificadas, inicia-se então o processo de obtenção.

Considerando a comunicação por meio do DDS, a distribuição dos dados de contexto ocorre da seguinte forma: os dados de contexto são encapsulados em uma estrutura de dados que é repassada à nuvem DDS que, por sua vez, é consumida pela aplicação ciente de contexto que tem interesse em receber determinadas informações de contexto.

5.3.3 Context Event Service

A distribuição consiste em publicar dados de contexto e notificações de eventos, a fim de que essas informações cheguem aos usuários locais ou remotos interessados. A versão original do CMS suporta apenas distribuição local, mas não remota. No MHNCS, ambos os tipos de distribuição são feitos via CDS. Se a publicação do dado não está condicionada a ocorrência de nenhum evento (cuja especificação é opcional), então o CDS simplesmente publica a informação de contexto. Se algum evento foi definido, então o CDS invoca o componente CES, que se encarrega de executar as regras contidas na descrição do evento, a fim de descobrir se esse evento ocorreu. A isso chamamos de validação de eventos.

O componente *Context Event Service* CES é responsável pela notificação baseada em eventos. Os eventos são regras que, quando consideradas verdadeiras, representam uma mudança de estado nas informações de contexto. Para tanto, os eventos podem ser definidos usando uma linguagem de definição baseada em XML. O esquema que detalha a estrutura da linguagem para definição de eventos é exposto no apêndice B.

Os eventos são registrados pelos consumidores, os quais definirão o interesse por determinada informação de contexto utilizando uma regra, neste caso a regra 5.3 foi definido de acordo com formalismo *Extended Backus-Naur Form* (EBNF) [35]. Neste caso, quando uma informação de contexto for atualizada, o CES irá validar de acordo com as condições definidas pelas aplicações ciente de contexto. Caso seja satisfeita, o CES notifica a aplicação.

```
1 <if_stmt> :: = if <exp> <stmt> else <stmt>
2 <exp> :: = <term> [{<connector> <term>}]
3 <term> :: = <variable> lt|eq|gt|lte|gte <value>
4 <connector> :: = and|or
5 <stmt> :: = true|false
```

Listagem 5.3: Descrição Formal da Regra de Validação de Eventos

A especificação de Eventos para relacionar várias regras e transformá-los em uma regra composta é necessário a definição de uma linguagem de composição de regras. Essa linguagem utiliza conectores para estabelecer conexão lógica entre regras simples. O CES implementa, além de eventos primitivos (simples), 2 (dois) tipos diferentes de relacionamentos cada um com seu respectivo conector. Com esses relacionamentos pode-se formar um grande número de expressões que formam regras compostas. Os relacionamentos são definidos a seguir:

a) **Simple**: define um evento simples também chamado de evento primitivo.

   - sem conector;
   - `systolicPressure > 180`.

b) **AND**: define dois ou mais eventos que devem acontecer independentemente de tempo ou ordem.

   - and;
   - `systolicPressure > 180 and diastolicPressure > 80`.

c) **OR**: define que um dos eventos ligados pelo conector deve acontecer.

   - or;
   - `systolicPressure > 180 or heartBeat >= 100`.

Para exemplificar o uso do CES, considere uma regra, definida na listagem 5.4, de emergência que indica que o SAMU deve ser notificado quando a pressão sistólica[2] do paciente for maior que 160mmHg e a pressão diastólica for maior que 80mmHg. Se o paciente for hipertenso, esse evento caracteriza que ele precisa de socorro. Caso a regra seja satisfeita, o CDS publica a notificação em um tópico DDS. Por questões de privacidade, apenas usuários autorizados a receber um dado de contexto conseguem se subscrever nesse tópico.

[2]Pressão arterial sistólica (PAS) é o maior valor verificado durante a aferição de pressão arterial. Exemplo: 120x80; onde 120 refere-se à pressão arterial sistólica e 80 refere-se à pressão arterial diastólica, ambas medidas em milímetros de mercúrio (mmHg)

```
<requirement>
  <event>
    <rule name="Have_high_blood_pressure">
      <condition context="systolicPressure" operator="gt" value="160"
          dataTypeValue="java.lang.Double" connector="and"/>
      <condition context="diastolicPressure" operator="gt" value="80"
          dataTypeValue="java.lang.Double"/>
    </rule>
  </event>
</requirement>
```

Listagem 5.4: Evento "*have high blood pressure*"

5.3.4 Implementando os Parâmetros de QoC no MHNCS

Para garantir a entrega das informações com um certo nível de qualidade implementamos quatro parâmetros de QoC com base nos requisitos discutidos no seção 5.1. Essa capacidade de atender requisitos de QoC na distribuição das informações é outra importante contribuição no trabalho proposto.

No MHNCS, algumas políticas de QoC são implementadas pelo DDS e outras pelo CDS. Para implementar os parâmetros frequência e taxa de atualização utilizamos políticas de QoS do DDS relacionadas à entrega de dados. Essa implementação não é trivial visto que em alguns casos é necessário combinar mais de uma política de QoS para atingir o parâmetro de QoC desejado. Já o parâmetro atualidade, depende do quanto os relógios dos dispositivos local e remoto estão sincronizados. Esse parâmetro é implementado pelo CDS. Já o parâmetro acurácia depende da informação provida pelos sensores.

Para implementar a *Frequency MINIMUM* foi utilizado a política de QoS *Deadline*, que garante que o CDS publicará pelo menos uma atualização no tópico DDS até o fim do intervalo; Para implementar a frequência *EXACT* utilizamos o *Deadline* e o *Time Based Filter*, que filtra as atualizações recebidas pelo *middleware* com base no tempo, eliminando qualquer outra atualização publicada pelo CDS que não seja a mais recente;

E por fim temos a frequência *MAXIMUM* que foi implementada semelhante à variante *MINIMUM*. Entretanto como a política *Time Based Filter* não pode ser configurada para permitir que mais de uma atualização passe pelo filtro, e as demais são descartadas pelo *middleware* DDS. Em função disso implementamos um *buffer* no componente CDS para armazenar as N últimas atualizações recebidas dentro de cada intervalo, descartando as excedentes. Então a aplicação utilizará o atributo *amount* para determinar a capacidade máxima de atualizações do *buffer*.

Foi Implementado o *Refresh rate Get_All* com as seguintes políticas de QoS: *History*, configurado como *Keep All*, que mantém um histórico de todas atualizações recebidas; *Destination Order*, que organiza o histórico em ordem decrescente em relação ao tempo de publicação (considerando que a última publicação é a mais recente); e *Latency Budget*, que estabelece um tempo mínimo entre o processamento das atualizações armazenadas. A cada intervalo, o histórico é esvaziado;

Com relação ao *Refresh rate Get_Last* implementamos de forma semelhante ao *Get_All*. A única diferença é que a política *History* é configurada como *Keep Last*. Essa configuração requer a definição do parâmetro *depth* da política *History*, que equivale ao tamanho do histórico e utilizamos atributo *amount* para determinar o tamanho do histórico.

Para prover a QoC *Freshness*, o MHNCS compara o *timestamp* do provedor, contido na notificação do dado de contexto, com o *timestamp* do consumidor registrado no ato do recebimento da notificação. A diferença entre os tempos equivale à idade da informação de contexto. Essa abordagem impõe que os relógios dos dispositivos produtores e consumidores de informações de contexto estejam sincronizados. Quanto menor for diferença de tempo entre os relógios, mais confiável será idade da informação. A sincronização não é garantida pelo MHNCS, de forma que apenas a idade de dados de contexto obtidos localmente podem ser determinadas de forma confiável pelo *middleware*.

Para prover os parâmetros de QoC *Accuracy*, os provedores de contexto do MHNCS fazem a busca por metadados que descrevam a acurácia aferidas pelo sensor em relação ao dado de contexto. Ao realizar a busca pelos provedores de contexto compatíveis com as especificações de QoC, o CDS selecionará a informação que tiver acurácia igual ou superior à exigida pelo usuário.

5.4 Considerações Finais

Neste capítulo foi apresentado o *middleware* **M**obile**H**ealth**N**et **C**ontext **S**ervice - MHNCS, cujo principal objetivo é controlar as principais tarefas de distribuição de dados entre aplicações móveis cientes de contexto em ambientes ubíquos. Foi discutido também neste capítulo o mecanismo de coleta e compartilhamento de dados entre os usuários móveis. Examinamos os principais detalhes do *middleware* desenvolvido com foco na área da saúde, permitindo a esses profissionais acompanhar o estado de saúde de seus pacientes diariamente sem a necessidade de encontros presenciais.

O objetivo do *middleware* proposto é gerenciar a distribuição dos dados de contexto e facilitar o desenvolvimento de aplicações móveis cientes de contexto. O MHNCS utiliza um modelo *publish/subscribe* centrado em dados para compartilhar esses dados entre as aplicações e com isso diminuir a complexidade da programação para ambientes móveis ubíquos. Além disso, o MHNCS permite a integração de diferentes aplicações móveis que necessitem de mecanismos de distribuição de dados de contexto. Essa integração foi obtida usando a especificação DDS, que define uma arquitetura centrada em dados, fornecendo interfaces padronizadas para publicação e subscrição de dados, garantindo assim um baixo acoplamento entre as aplicações que produzem dados de contexto e seus respectivos consumidores.

Como principais contribuições do MHNCS podemos destacar a modificação do *middleware Context Management Service* e o desenvolvimento de componentes para especificação, solicitação, obtenção, análise e distribuição de dados de contexto, considerando requisitos de qualidade das informações (parâmetros de QoC). A unidade básica do *middleware* proposto é o componente de distribuição de contexto, *Context Distribution Serivce* (CDS), responsável por distribuir toda as informações de contexto de acordo com os interesses registrados anteriormente pelas aplicações cientes de contexto. Além disto, para facilitar a definição dos requisitos de contexto foi proposta uma linguagem, a *MHN Context Language*, que permite as aplicações determinar os requisitos de contexto que têm interesse em receber esses dados, deixando transparente os mecanismo de aquisição e seleção dos provedores de contexto. Outro ponto importante que devemos mencionar é o mecanismo de notificação baseado em eventos que é uma importante característica no MHNCS. Esse

mecanismo permite que as aplicações possam definir regras para que sejam notificados de acordo com a mudança do ambiente monitorado, e isso é obtido pela filtragem dos dados de determinados tópicos a partir de seus atributos e valores específicos, com isso reduzir o tráfego de mensagens entre as aplicações móveis solicitantes.

6 TRABALHOS RELACIONADOS

Na literatura encontram-se vários trabalhos relacionados à distribuição de dados de contexto [8]. No entanto, a grande maioria dos trabalhos não aborda questões relacionadas a como atender requisitos de QoC, algo que apenas recentemente tem sido explorado. Este capítulo relata de forma sucinta alguns sistemas de *middleware* e *framework* voltados para distribuição de dados de contexto. Ao descrever cada *middleware*, fazemos uma comparação com o MHNCS, a fim de reforçar nossas contribuições.

6.1 Health Support in Aware and Ubiquitous Domestic Environments

O *Heath Support in Aware and Ubiquitous Domestic Environments* (H-SAUDE), descrito por Copetti et al. [17], é um *framework* que tem o objetivo de realizar a distribuição de informações para aplicações móveis ciente de contexto aplicados na área da saúde. O *framework* H-SAUDE realiza o monitoramento dessas informações referentes ao ambiente (e.g. umidade e temperatura), comportamento (e.g. movimento, parado e realizando atividade física) e fisiológicas (e.g. pressão sistólica - PAS, pressão diastólica - PAD e frequência cardíaca) dos pacientes.

Esse trabalho propõe um acompanhamento do estado de saúde dos pacientes baseado em lógica nebulosa para identificar situações críticas caracterizadas principalmente por alterações no sinais vitais dos pacientes. O *framework* obtém os dados a partir de sensores que estão localizados no corpo do paciente e/ou em sua residência e, por meio de regras baseadas no conhecimento de especialistas (profissionais da saúde), realiza a inferência destas informações, gerando alarmes para um centro de acompanhamento de pacientes, caso ocorram desvios na normalidade do estado de saúde do paciente. Os dados obtidos dos sensores são transmitidos por uma rede sem fio para as aplicações móveis ciente de contexto e também para o centro de acompanhamento dos pacientes para o processo de inferência dos dados coletados.

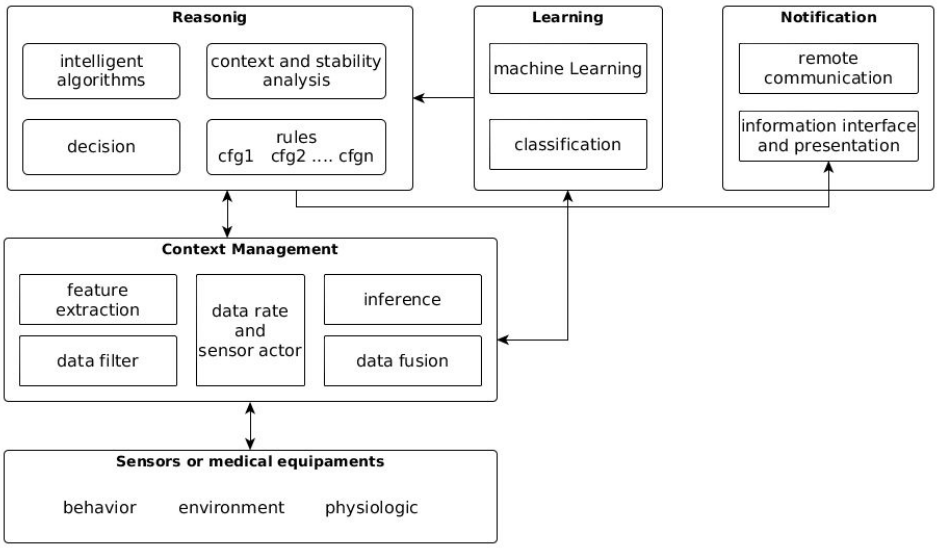

Figura 6.1: Arquitetura do H-SAUDE.

Fonte: Copetti et al. [17]

A arquitetura do H-SAUDE é dividida em quatro módulos, conforme mostra a Figura 6.1: o gerenciamento de contexto (*Context Management*), de raciocínio (*Reasoning*), de aprendizado (*Learning*) e de notificação (*Notification*). O módulo gerenciamento de contexto gerencia os dados dos sensores buscando obter as informações relevantes sobre o estado de saúde do paciente. O módulo de raciocínio é o núcleo do *framework* e possibilita que uma situação crítica, baseada no pré-processamento dos dados obtidos dos sensores, possa produzir alertas para a aplicação ciente de contexto. O módulo de aprendizado tem o objetivo de individualizar parâmetros de referência utilizados para cada paciente. A base de treinamento pode ser formada por exames realizados previamente pelo paciente.

A classificação do estado do paciente é realizada no módulo de raciocínio e possui os seguintes estados possíveis: **normal**, **alerta** e **emergência**. O estado **normal** significa que o paciente está bem, sem necessidade de intervenção dos serviços. O estado **alerta** significa que o monitoramento deverá ser ajustado para observar melhor o estado do paciente, por exemplo, diminuir o tempo de monitoramento das variáveis fisiológicas. O estado de **emergência** significa que alertas deverão ser disparados para serviços de emergência. Por último, o modulo de notificação que realiza a comunicação tanto com os usuários quanto com uma central de monitoramento. O seu princípio básico é que o sistema deve enviar notificações conforme as mudanças dos dados de contexto. Dependendo do tipo de evento ocorrido e do estado de saúde do paciente,

o sistema utiliza um conjunto de regras que são associadas aos dados de contexto (*e.g.* ambientais, comportamentais e fisiológicos), tornando os alertas mais confiáveis para cada situação.

6.2 MobiCon

Mobicon [32] é um *middleware* de monitoramento de contexto que é capaz de lidar com problemas e desafios de recursos em ambientes ubíquos ricos em sensores. Sua arquitetura consiste em uma parte central embarcada em dispositivos móveis que se comunica com uma série de sensores externos ao dispositivo e espalhados no ambiente. Essa infraestrutura possui uma abordagem de controlar o nível de energia dos sensores de acordo o interesse das aplicações (expressos por meio das consultas).

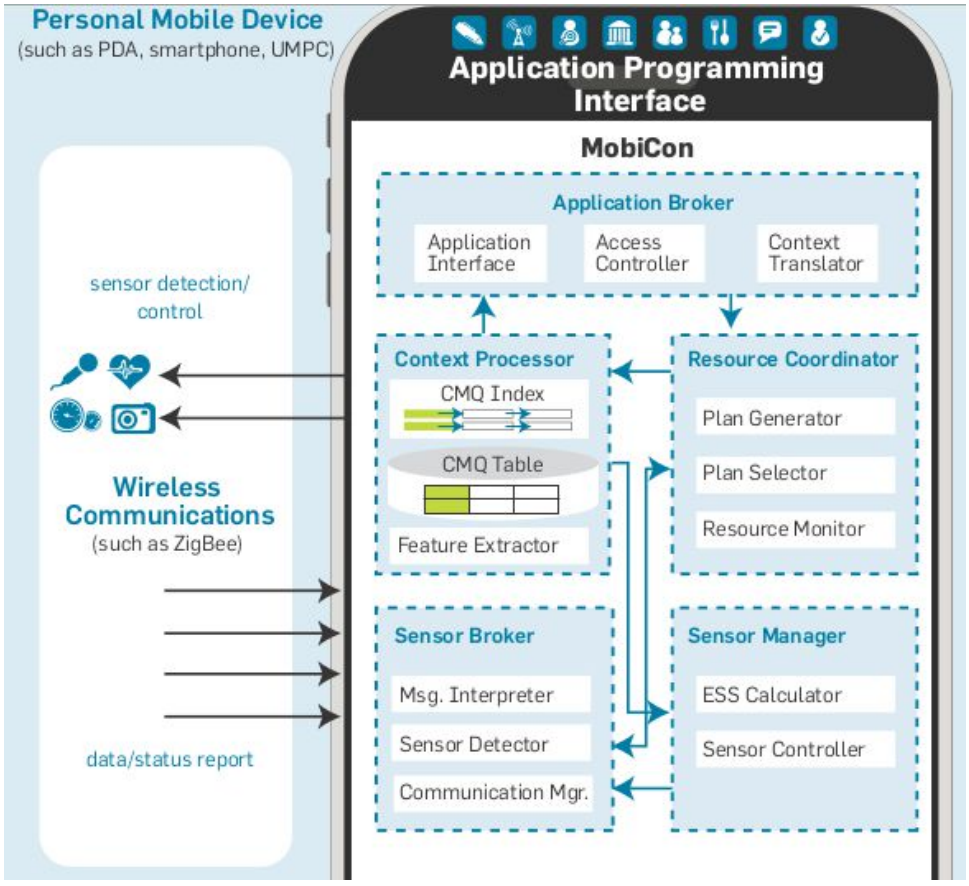

Figura 6.2: Arquitetura do MobiCon.

Fonte: Lee et al. [32]

O MobiCon possui cinco componentes: *Context Processor*, *Sensor Manager*, *Resource Coordinator*, *Application Broker* e *Sensor Broker*. O *Context Processor* gerencia a

distribuição dos dados de contexto, avaliando os dados entregues pelo *Sensor Broker* de acordo com as consultas definidas pela aplicação ciente de contexto.

O *Sensor Manager* realiza uma busca por um conjunto mínimo de sensores que disponibilizam as informações de contexto requisitadas por cada aplicação. O componente *Resource Coordinator* gerencia o nível de energia dos sensores que estão sendo utilizados e avalia em tempo de execução a necessidade de trocar um determinado sensor por outro que tenha maior nível de energia.

A definição dos requisitos de contexto é feita por meio de requisições, e essas requisições são realizadas a partir de um registro de consultas de alto nível chamado *Context Monitoring Query* (CMQ). As CMQ serão posteriormente traduzidas para uma expressão lógica de baixo nível, por exemplo a CMQ `temperature == hot` poderia ser traduzida para `thermometer > 86`. A partir da expressão de baixo nível e de regras de expressões booleanas, o *middleware* encontra um conjunto mínimo de sensores necessários para avaliar as CMQ. Apenas esse conjunto mínimo de sensores necessários é utilizado, enquanto os outros sensores são ou serão desativados, possibilitando economia de recursos.

O fato do funcionamento do MobiCon ser baseado em consultas de alto nível, não permite que aplicações que o utilizem tenham acesso aos valores de contexto de baixo nível (*raw data*) sem a definição de CMQs. Por exemplo, uma aplicação não é capaz de receber dados de contexto continuo do termômetro, apenas saber se a temperatura é considerada alta ou não.

6.3 Mobilis Dresden

O Mobilis [37] [52] é um *middleware* para suporte ao desenvolvimento de aplicações de RSMs que disponibiliza serviços para o compartilhamento das informações de contexto dos usuários, gerenciamento de grupos, serviços de armazenamento de arquivos e meta-dados a eles associados, bem como a edição colaborativa de textos e imagens. Esse *middleware* possui a arquitetura ilustrada na Figura 6.3.

O *middleware* tem sua comunicação baseada no protocolo *Extensible Messaging and Presence Protocol* (XMPP) [55] [48]. Todos os serviços Mobilis têm os seus

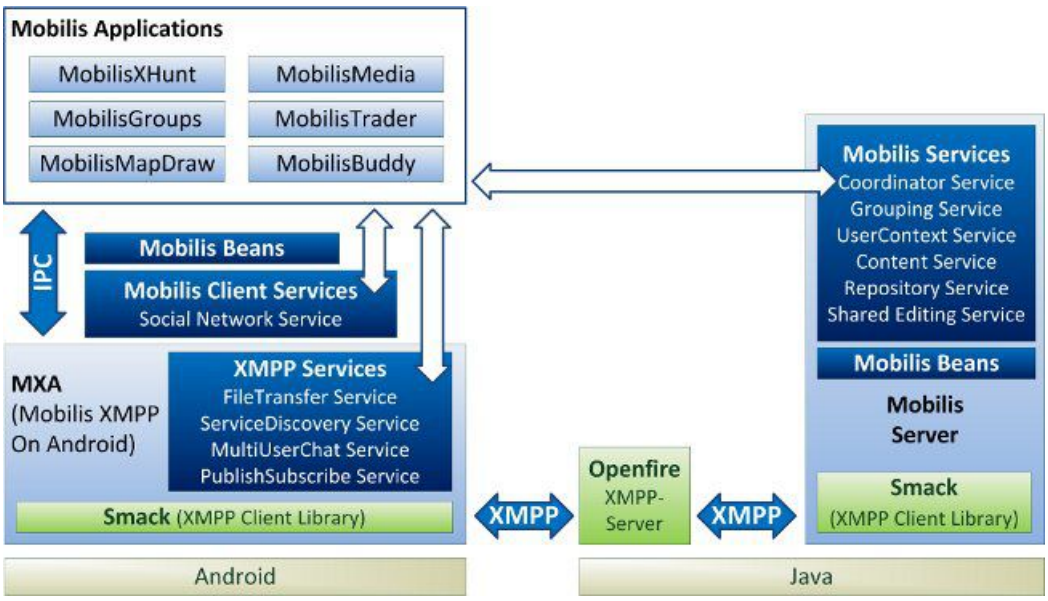

Figura 6.3: Arquitetura do Mobilis.

Fonte: Lübke [37].

próprios identificadores XMPP que são utilizados para troca de mensagens utilizando-se este protocolo. A biblioteca cliente *XMPP Smack* é usada para a comunicação entre as aplicações móveis cientes de contexto. Para a troca de mensagens XMPP, o Mobilis utiliza o servidor XMPP *Openfire*.

O componente *Mobilis Beans* compreende uma coleção de estruturas que ajudam a criar os diversos pacotes XMPP que são trocados entre o servidor Mobilis e os clientes. Estes podem ser utilizados por desenvolvedores para realizar solicitações para os serviços que executam no lado do servidor. O *MobilisServer* ou servidor Mobilis oferece diversos serviços para as aplicações clientes. O Coordenador de Serviço (*Coordinator Service*) é responsável por gerenciar todos os serviços oferecidos pelo servidor Mobilis, o que inclui a criação de serviços. A consulta por serviços é feita para o coordenador de serviço que possui um mecanismo de descoberta de serviços (*Service Discovery*) que retorna um identificador único do serviço solicitado, se este existir.

O Serviço de Contexto do Usuário (*User Context Service*) gerencia todas as informações de contexto dentro da plataforma Mobilis, tais como informações de contexto físico (como luminosidade e temperatura), contexto técnico (como largura de banda e taxa de erro), contexto pessoal (como endereço e nome) podem ser adquiridos a partir deste serviço. O *middleware* Mobilis propõe apenas tratar as informações de

contexto e fornecê-las às aplicações. Segundo o modelo proposto, a infraestrutura não é focada na obtenção da informação, ou seja, sua aquisição da informação junto aos sensores. A ênfase é o tratamento e entrega de dados já processados para as aplicações. Os sensores registram-se no módulo *User Context Service* e publicam as informações através de um fluxo de dados XMPP. As informações provenientes dos sensores são processadas e depois são entregues às aplicações de acordo com os registro feito por elas.

O Serviço de Grupo (*Grouping Service*) é um serviço que tem como base a criação de grupos de acordo com a localização dos usuários, ou seja, é possível a criação e a gestão de grupos com restrições específicas (temporal e local, por exemplo) a partir de sua localização e até a união de grupos.

O Serviço de Conteúdo (*Content Service*) e o Serviço de Repositório (*Repository Service*) tem como principal tarefa o armazenamento de mídias e meta-dados a elas associadas, respectivamente. O Serviço de Edição Colaborativa (*Collaborative Editing Service*) permite o processamento conjunto e simultâneo de dados por várias pessoas distribuídas geograficamente em um processo de edição compartilhada ou colaborativa.

O MXA (*Mobilis XMPP on Android*) encapsula as funcionalidades da biblioteca *XMPP Smack* e promove uma interface AIDL (*Android Interface Definition Language*) que é utilizada em aplicações Mobilis para se comunicar com o servidor que oferece os diversos serviços descritos anteriormente.

6.4 MobiSoc

O MobiSoc [26] [10] provê uma plataforma que possibilita o desenvolvimento de aplicações sociais móveis cientes de contexto capazes de capturar, gerenciar e compartilhar informações sobre o estado social das comunidades em que o usuário se encontra fisicamente. Esse estado é composto de informações sobre os perfis dos usuários e de lugares, além de conter informações relacionadas à afinidade entre pessoas e afinidades que pessoas possuem com lugares.

O estado dessa rede social evolui continuamente ao longo do tempo com a criação de novos perfis de usuário, laços sociais, informações relacionadas a lugares

ou eventos. O MobiSoc inclui também algoritmos de aprendizagem para descobrir padrões geo-sociais previamente desconhecidos, tais como afinidades entre pessoas e entre pessoas e lugares.

A arquitetura do MobiSoc é apresentada na Figura 6.4. O MobiSoc atua como uma entidade centralizada para o gerenciamento da rede social e fornece uma API de serviço para desenvolvimento de aplicações. A arquitetura é dividida em módulos e sub-módulos. Os módulos internos podem ser fisicamente distribuídos, a fim de prover a escalabilidade para vários clientes móveis.

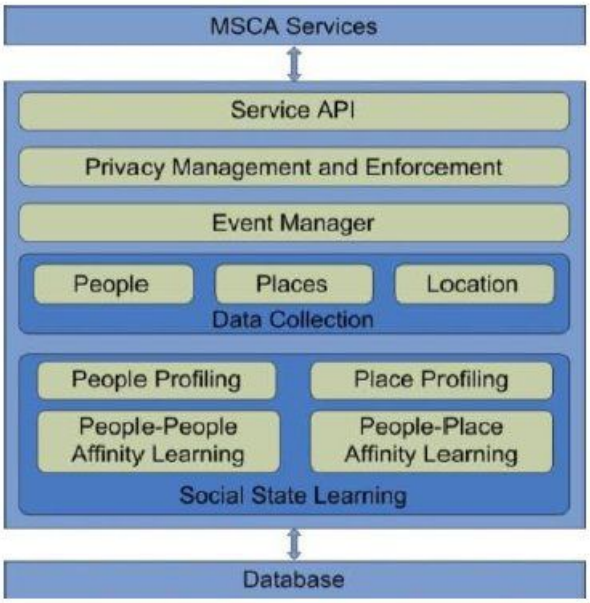

Figura 6.4: Arquitetura do MobiSoc.

Fonte: Borcea et al. [10]

Os módulos que compõe o MobiSoc são: Coleção de Dados (*Data Collection*), Aprendizagem de Contexto Social (*Social State Learning*), Gerenciador de Eventos (*Event Manager*) e Gerenciador de Privacidade e Execução (*Privacy Management and Enforcement*). O módulo Coleção de Dados é composto por três sub-módulos: Pessoas (*People*), Lugares (*Places*) e Localização (*Location*). O submódulo Pessoas permite às aplicações coletarem, armazenarem e modificarem as informações dos perfis dos usuários (como interesses, preferências, entre outras). O sub-módulo Lugares armazena informações de dados geográficos sobre construções e mapas. Além disso, provê mecanismos para adicionar informações sobre eventos relacionados com o lugar. Por fim, o submódulo Localização recebe e armazena as atualizações das informações de localização provenientes dos dispositivos móveis dos usuários.

Aprendizagem de Contexto Social é responsável por inferir novas informações sobre o relacionamento entre pessoas e entre pessoas e lugares. Este módulo é composto pelos seguintes sub-módulos:

- Perfil do Usuário (*People Profiling*): provê informações sobre o perfil do usuário, grafo das conexões sociais deste e os grupos sociais que faz parte;
- Perfil do Lugar (*Place Profiling*): compartilha informações armazenadas sobre o lugar e aprimora a semântica do lugar com informações sociais;
- Aprendizagem de Afinidade Pessoa-Pessoa (*People-People Afinity Learning*): responsável por computar afinidades sociais entre pares de usuários, com base nos interesses pessoais, amigos em comum, ou ainda, lugares em comum que costumam frequentar;
- Aprendizagem de Afinidade Pessoa-Lugar(*People-Place Afinity Learning*): responsável por descobrir quais são os lugares de interesse dos usuários, através da análise histórica dos dados de localização dos mesmos.

O Gerenciador de Eventos é o módulo utilizado para comunicação assíncrona com aplicações. Aplicações podem registrar eventos no *middleware* para receber notificações quando algo de interesse do usuário ocorrer na rede social. Por exemplo, um usuário pode desejar uma notificação quando um outro usuário específico acessar a rede. O dispositivo móvel realiza periodicamente a busca de notificações de eventos.

O Gerenciador de Privacidade e Execução é o modulo que gerencia as regras de privacidade e acesso das entidades no sistema (usuários e aplicações). Tais regras são armazenadas em uma base de dados. Cada aplicação registra suas preferências de acesso e privacidade. Estas regras são expressas em forma de sentenças que contém uma entidade primaria e uma secundária, esta por sua vez, pode ser um usuário ou um grupo.

O MobiSoc possui sua arquitetura orientada a serviço (SOA). Seus serviços são acessados utilizando kSOAP[1], um *framework Simple Object Acess Protocol* (SOAP)[2] é um protocolo para troca de informações estruturadas em XML e é utilizado no desenvolvimento de aplicações para serviços *web* para J2ME. O kSOAP vem de kXML, um XML utilizado em pequenos dispositivos com recursos limitados, em especial de memória e processamento.

O SOAP foi escolhido devido a sua portabilidade, já que oferece uma independência de linguagem e, somado a isto, existem clientes SOAP em várias linguagens populares. Além disso, sua comunicação entre clientes e servidores se dá a partir do protocolo HTTP.

Existe uma grande variedade de sensores responsáveis por alimentar o *Data Collection* com informações relevantes. Esses sensores, entretanto, muitas das vezes têm interface de consulta heterogênea. É necessária uma maneira de disponibilizar as informações de forma que seu acesso seja de fácil utilização pelas aplicações. Para disponibilizar as informações, o Mobisoc utiliza uma base de dados, aproveitando a vantagem da padronização de acesso aos dados.

Neste caso, as fontes de contexto têm que realizar um registro no módulo *Data Collection*. E cada sub-módulo do *Data Collection* tem a responsabilidade de armazenar, processar e entregar as informações para cada aplicação. Semelhante ao Mobilis, o Mobisoc também não é focada na obtenção da informação, seu foco é o tratamento e entrega de dados às aplicações.

6.5 OpenCOPI

OpenCOPI [34] é uma plataforma para integração de diferentes *middleware* de distribuição de contexto que fornece um serviço de contexto unificado e baseado em ontologias, bem como um ambiente que facilita o desenvolvimento das aplicações ubíquas pela definição de *workflows* semânticos com a descrição abstrata da aplicação. Os autores definem *workflow* como descrição do fluxo de atividades para se alcançar a

[1]MCHUGH, Jeff.Low Bandwidth SOAP. Disponivel em:<http://www.xml.com/pub/a/ws/2003/08/19/ksoap.html>. acesso em janeiro de 2014

[2]SOAP Specifications - World Wide Web Consortium. Disponivel em:<http://www.w3.org/TR/soap/>. acesso em janeiro de 2014

meta ou objetivo das aplicações cientes de contexto. O objetivo principal do OpenCOPI é facilitar a tarefa de desenvolver e executar aplicações ubíquas e cientes de contexto. Para alcançar esse objetivo, o OpenCOPI integra diferentes provedores de serviços usados pelas aplicações. Dentre esses provedores incluem-se as aplicações de provisão de contexto. Essa plataforma permite que diferentes provedores colaborem entre si para alcançar um objetivo de alto nível, provendo serviços com valor agregado e informações de contexto para aplicações ubíquas, disponibilizando as seguintes funcionalidades:

- Mecanismo de composição automática de serviços baseado nas preferências do usuário e nos metadados dos serviços;
- Modelos de contexto e de comunicação padronizados;
- Mecanismo de adaptação que é disparado quando ocorrem falhas na execução de algum serviço utilizado por uma aplicação ou em caso de mudanças na qualidade dos serviços usados por uma aplicação.

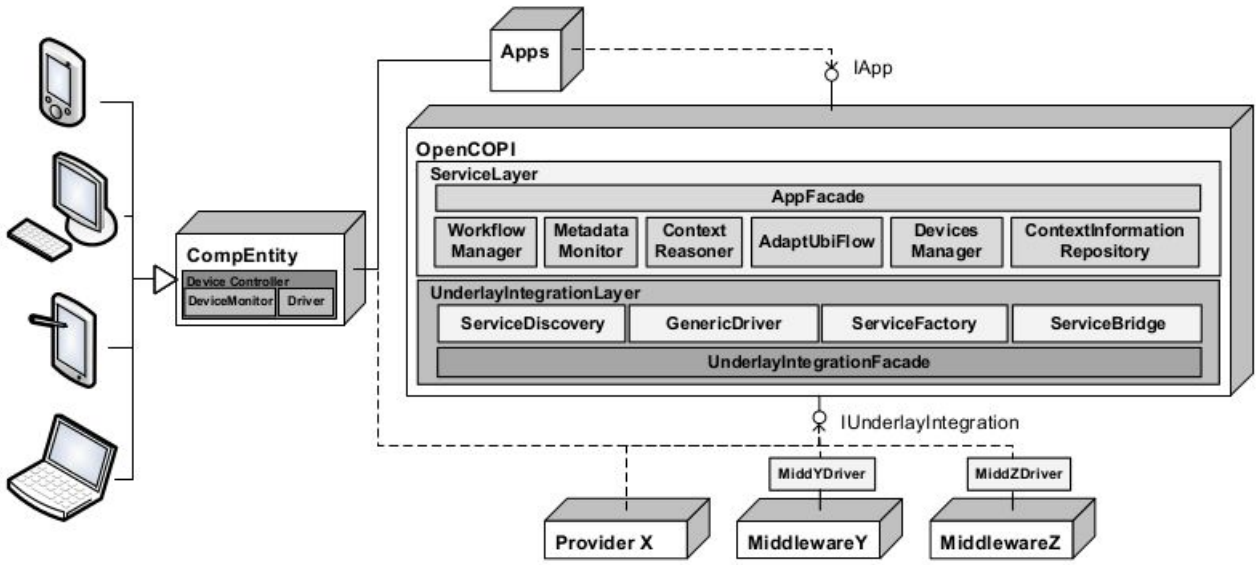

Figura 6.5: Arquitetura do OpenCOPI.
Fonte: Lopes et al. [34].

A arquitetura do OpenCOPI é dividida em duas camadas, a *ServiceLayer* e a *UnderlayIntegrationLayer*, e possui duas interfaces, a *IApp* e *IUnderlayIntegration*, conforme a Figura 6.5. A camada *ServiceLayer* é responsável por gerenciar os serviços oferecidos pelos provedores de serviços que são representados pela ontologia *Web*

OntologyLanguage for Web Services (OWL-S). Os componentes dessa camada usam as abstrações dos serviços para a composição de serviços, criação e seleção dos planos de execução e para realizar adaptações. Os componentes dessa camada também suportam o armazenamento e inferência de contexto, entre outras funcionalidades.

A interface *IApp* oferecida pelo OpenCOPI, permite que as aplicações possam criar e invocar os *workflows* semânticos. A camada *UnderlayIntegrationLayer* é responsável por integrar os provedores de serviços e de contexto através da conversão do modelo de contexto das plataformas subjacentes para o modelo de contexto do OpenCOPI. Além disso, é necessário abstrair o protocolo de comunicação das plataformas subjacentes quando estas não adotarem os protocolos padronizados dos serviços Web, mas sim outros protocolos (e.g. Sockets, RMI, CORBA). A interface *IUnderlayIntegration* interliga os provedores de serviços e a camada *UnderlayIntegrationLayer* do OpenCOPI.

6.6 Análise Comparativa

Devido à grande atenção que a computação ubíqua tem recebido nos últimos anos, várias pesquisas vêm sendo realizadas, resultando em um grande número de plataformas com o objetivo de facilitar o desenvolvimento de aplicações ubíquas. Um aspecto importante dessas plataformas tem sido definir mecanismos para prover informações de contexto para as aplicações ubíquas, tornando-as conhecidas também como *middleware* de provisão de contexto.

A Tabela 6.1 ilustra as características analisadas para comparação entre o *middleware* MHNCS e as plataformas citadas no neste capítulo. São analisados aspectos relacionados ao modelo de representação, qualidade de contexto, mecanismo de distribuição, especificação de comunicação, gerenciador de eventos e linguagem de especificação de contexto.

- Modelo de representação: permite analisar o modelo que representa os dados de contexto que permite uma interação padronizada entre as aplicações;
- Mecanismo de distribuição: requisito utilizado para analisar os paradigma de comunicação;

- Especificação de comunicação: analisa os modelos e técnicas que possam prover uma comunicação distribuída e transparente; e analisa o mecanismo de processamento das mensagens das plataformas;

- Suporte a eventos: analisa o suporte de notificação baseado em eventos entre as plataformas;

- Suporte a qualidade de contexto: avalia se a infraestrutura tem suporte à análise da qualidade da informação;

- Especificação de requisitos: visa analisar os mecanismos que facilitem a especificação de requisitos de contexto.

Plataforma	Modelo de representação	Mecanismo de distribuição	Especificação de comunicação	Suporte a Eventos	Suporte a qualidade de contexto	Especificação de requisitos
H-SAUDE	Ontologia	*Publish/Subscribe*	n/d[3]	Sim	Não	Método
Mobicon	Chave-Valor	*Publish/Subscribe*	n/d	Sim	Não	Método
Mobilis	Linguagem de Marcação	*Publish/Subscribe*	XMPP	Não	Não	Método
Mobisoc	Esquema de Marcação	*Publish/Subscribe*	SOAP	Sim	Não	Método
OpenCOPI	Ontologia	*Publish/Subscribe*	SOAP	Não	Sim	Linguagem de descrição
MHNCS	Orientado a Objetos	*Publish/Subscribe*	DDS	Sim	Sim	Linguagem de descrição

Tabela 6.1: Análise Comparativa dos Trabalhos Relacionados.

[3]n/d: Especificação de comunicação não foi definido pelos autores

O modelo de representação é um requisito muito importante uma vez que descreve como as informações de contexto serão disponibilizadas para as aplicações. As plataformas H-SAUDE e OpenCOPI usam um modelo baseado em ontologia, que permite, entre outras coisas, um compartilhamento de contexto mais fácil principalmente pelo seu poder de relacionar as informações de contexto entre si, possibilitando o uso de máquinas de inferências para deduzir informações de contexto complexas a partir de informações de contexto mais simples. Entretanto, o modelo baseado em ontologia exige um elevado nível de processamento e os trabalhos que utilizam esse modelo acabam delegando para a nuvem o mecanismo de análise de contexto.

Com relação ao modelo chave-valor, utilizado na plataforma Mobicon, é um modelo de representação mais simples para representar os dados de contexto. Esse modelo não permite a inclusão de qualquer tipo de hierarquia entre os atributos e, apesar de ser um modelo bastante simples de utilizar e manipular, não permite estruturações mais sofisticadas que habilitem algoritmos eficientes de recuperação de contexto. A característica principal do modelo baseado em marcação é uma estrutura de dados hierárquica que contém rótulos de marcação com atributos e conteúdo, como visto na plataforma Mobisoc. Esse modelo tende a gerar descrições maiores que os de formatos binários e consomem maior largura de banda da rede e espaço de armazenamento, ou exigem maior tempo de processamento para compressão.

O modelo orientado a objetos, utilizado em nossa proposta, permite organizar melhor os dados de contexto, e simplificar a criação de regras de manipulação do contexto. As habilidades de herança e reuso simplificam a representação do conhecimento, a definição de propriedades, definição de regras, domínios complexos e não exige um elevado nível de processamento. Além disso, esse modelo permite que os mecanismo de análise de contexto sejam embarcados no próprio dispositivo móvel.

Outro ponto observado nas plataformas de distribuição de contexto é o provimento de informações de contexto através do mecanismo *Publish/Subscribe*. Esse mecanismo está recebendo cada vez mais atenção por fornecer um baixo acoplamento entre as aplicações cientes de contexto e os produtores de contexto. Essa característica permite aumentar o suporte à interoperabilidade, mobilidade e transparência para as aplicações ubíquas.

Com relação à especificação de comunicação, a plataforma Mobilis utiliza a especificação XMPP e Mobisoc e OpenCOPI utilizam a SOAP para troca de mensagens entre as aplicações. Ambas as especificações garantem as aplicações um desacoplamento de sincronização e de espaço, permitindo que não haja um bloqueio durante a comunicação entre as aplicações e as fontes de contexto, podendo obter de forma assíncrona as notificações. A desvantagem dessas especificações é que não garantem um desacoplamento de tempo [22] entre as aplicações.

Para garantir o desacoplamento de tempo em nossa proposta utilizamos a especificação DDS que é baseado no modelo DCPS. Isso possibilitará que as aplicações não precisem estar participando ativamente da interação ao mesmo tempo. Em particular, a fonte de contexto pode publicar alguns acontecimentos enquanto a aplicação cliente estiver desconectado. Essa característica é útil para aplicações móveis, cuja comunicação está sujeita à conectividade intermitente e eventuais desconexões.

Analisando o suporte de eventos das plataformas, observamos que a plataforma H-SAUDE utiliza lógica *Fuzzy* para avaliar as informações de contexto e identificar situações críticas. Já a Mobisoc utiliza regras expressas em lógica de primeira ordem para avaliar as situações de cada usuário de acordo com as informações disponíveis. Tanto H-SAUDE e Mobisoc não utilizam uma linguagem de definição de eventos.

Diferentemente da H-SAUDE e Mobisoc, a plataforma MobiCon utiliza uma linguagem chamada CMQ que visa facilitar a definição dessas regras, expressas por meio de lógica de primeira ordem. Semelhante ao Mobicon, o MHNCS utiliza uma linguagem, chamada de *Context Event Language*, que permite às aplicações ubíquas receber as informações de contexto envolvidas nas constantes mudanças em seu estado de execução, considerando o ambiente altamente dinâmico em que executam.

Aplicações ubíquas são inerentemente dinâmicas, uma vez que usam dispositivos móveis que podem entrar e sair frequentemente da área de abrangência de uma dada rede e também estão sujeitas a eventuais desconexões. Nesse contexto altamente dinâmico, as aplicações necessitam que, durante sua execução, as informações de contexto que elas usam mantenham níveis de qualidade que satisfaçam seus requisitos. Como podemos observar na Tabela 6.1, o suporte a qualidade de

contexto é outra característica que tem sido abordada recentemente em trabalhos na área da computação ubíqua.

O OpenCOPI utiliza QoC para selecionar os serviços e as informações disponíveis no ambiente ubíquo. OpenCOPI recebe um plano de execução em que cada parâmetro de QoC pode ser configurado com diferentes graus de importância variando entre 0(zero) e 1(um). Desse modo, um determinado parâmetro que não seja levado em consideração na seleção é definido com grau zero, enquanto que um dado parâmetro que seja o único considerado na seleção deve ter o grau de importância definido como um. No OpenCOPI não foi observado um mecanismo que permita às aplicações controlar o número de mensagens recebidas por meio dos parâmetros de QoC, que é um requisito essencial, como descrito no capítulo 5, no desenvolvimento de aplicações ciente de contexto.

Em nossa proposta usamos os parâmetros de QoC para analisar e selecionar as informações com certo nível de qualidade, satisfazendo os requisitos das aplicações. Além disso, usamos parâmetros de QoC para controlar o fluxo de entrega das informações, garantindo que as aplicações irão receber apenas informações que deseja processar a cada intervalo. Essa garantia possibilita uma economia de processamento e de energia, proporcionando um aumento no tempo de vida dos dispositivos móveis. Essa característica é obtida utilizando as políticas de qualidade de serviço disponibilizada pela especificação DDS.

Em vários dos trabalhos analisados a definição dos requisitos de contexto por parte da aplicação é realizada por intermédio de chamadas de métodos disponibilizadas pelos *middlewares*. Para facilitar a definição dos requisitos de contexto por parte da aplicação, OpenCOPI utiliza uma linguagem baseada em XML que permite a descoberta de dados de contexto oferecidos por vários provedores de contexto. De forma semelhante, em nossa proposta também definimos uma linguagem baseada em XML, que chamamos de *MHN Context Language*. Consideramos nossa linguagem bastante expressiva. Ela permite a definição dos requisitos de forma fácil e rápida por parte dos desenvolvedores das aplicações sem a necessidade de inserir código para realizar chamadas a métodos que definem os requisitos de contexto requeridos pela aplicação.

Em termos comparativos, acreditamos que o MHNCS suporta um conjunto mais amplo de funcionalidades voltadas para o desenvolvimento de aplicações móveis cientes de contexto que as plataformas apresentados neste capítulo. Além de uma linguagem flexível que dá suporte à especificação de QoC, a privacidade dos dados de contexto é garantida por meio de mecanismos de segurança e privacidade do MobileHealthNet. A obtenção dos dados é facilitada pela busca e instanciação dinâmica de provedores de contexto realizada pelo CMS. As regras podem ser validadas pelo MHNCS , a fim de garantir a entrega e notificações quando há a ocorrência de eventos. Por conta do uso do DDS, o MHNCS provê a distribuição de dados de contexto considerando políticas de QoS que ajudam a prover QoC, e também permite diferentes configurações de confiabilidade na entrega dos dados de contexto. Utilizando a especificação DDS nossa infraestrutura possibilita um fraco acoplamento entre as aplicações, garantindo a entrega das mensagens, sem requerer qualquer intervenção das aplicações no processo de distribuição desses dados. Essas características permitem ao desenvolvedor se preocupar somente com o conteúdo que será transmitido entre as aplicações.

7 AVALIAÇÃO

Um sistema de coleta, distribuição e análise de dados de contexto, como o MHNCS, impõe um custo computacional nos dispositivos no qual ele executa. A preocupação com este custo é ainda maior quando se trata de dispositivos móveis, que possuem baixo poder de processamento e pouca memória. Dessa forma, é necessário que o custo gerado pelo MHNCS seja o menor possível de modo que não comprometa o desempenho do sistema ao ponto de tornar inviável o seu uso. Diante disso, após o desenvolvimento da infraestrutura de distribuição ciente de contexto para o projeto MobileHealthNet, foram realizados alguns testes com o intuito de avaliar o impacto causado pelo *middleware* no ambiente de execução.

Além disso, realizamos uma avaliação qualitativa sobre as funcionalidades do MHNCS através de questionários respondidos por um grupo de desenvolvedores de software que utilizaram o *middleware* para o desenvolvimento de aplicações. As informações obtidas nesse processo de avaliação qualitativa são tão importantes quanto a desempenho do sistema, não podendo ser obtidas de outra forma senão pelo questionamento dos utilizadores do *middleware*. Neste capítulo, descreveremos como o *middleware* proposto nesta dissertação foi avaliado.

7.1 Avaliação Quantitativa

Para definirmos os valores dos parâmetros utilizados nos experimentos realizados para a avaliação quantitativa do MHNCS, tomamos como base um cenário fictício mas que ilustra bem um uso típico previsto para o *middleware*. Nesse cenário, uma equipe de profissionais de saúde deseja receber informações de um paciente portador de doença crônica diariamente, sem a necessidade de encontro presenciais. Para suprir essa necessidade, criamos três provedores de contexto que irão publicar os dados dos pacientes. Essa aplicação precisa estar ciente das seguintes informações de contexto do paciente: batimento cardíaco, frequência respiratória e localização.

As duas primeiras podem ser obtidas pelo MHNCS a partir do sensor *Zephyr BioHarness 3* [1], que envia os dados de contexto ao dispositivo móvel do paciente por meio de uma conexão *bluetooth*; já a localização, é obtida a partir do GPS do dispositivo móvel.

O dispositivo móvel utilizado nos testes tem as seguintes configurações: processador Cortex-A8 1GHz, 512MB de RAM, Sistema Operacional Android[2] versão 2.3.3. Esse dispositivo possui ainda, além da arquitetura WPAN, um suporte WLAN, rede utilizada para publicar os dados de contexto na nuvem DDS.

Nesta avaliação quantitativa, observamos que o consumo de memória e processamento por parte do MHNCS no dispositivo do paciente monitorado. Considerado de baixo poder computacional, esse dispositivo possui uma configuração compatível àquela dos potenciais pacientes que serão beneficiados pelo do MobileHealthNet, já que a população alvo desse projeto compreende comunidades carentes e distantes.

O objetivo desse teste é avaliar se o nosso *middleware* realmente pode ser executado com desempenho satisfatório em dispositivos com baixo poder computacional. Os profissionais de saúde que recebem os dados de contexto podem usar *tablets* ou *smartphones* com maior capacidades, ou ainda computadores pessoais com a plataforma Android. Por esse motivo, não foi realizado a avaliação do consumo de recursos do MHNCS nesses dispositivos, que normalmente apenas recebem os dados de contexto.

O teste quantitativo consistiu em mensurar o consumo de processamento e memória exigidos pelos componentes de obtenção CMS, validação de eventos CES e distribuição dos dados de contexto CDS, variando a quantidade de provedores em execução simultânea em cada experimento entre 0 e 3. Essa variação teve por objetivo analisar o impacto no incremento progressivo de provedores de contexto em execução no consumo de recursos.

Cada provedor de contexto obtinha a informação do sensor a cada 1 segundo. Na Tabela 7.1 apresenta os resultados da avaliação quantitativa. Cada

[1] http://www.zephyranywhere.com/products/bioharness-3/

[2] http://developer.android.com/index.html

experimento teve duração de 30 minutos. Para a obtenção dos resultados foi utilizada a ferramenta *Dalvik Debug Monitor Server* (DDMS)[3].

Componente	**Número de Provedores de Contexto**			
	0	**1**	**2**	**3**
CDS	≅0 / 14,27	3,2 / 14,35	3,4 / 14,36	3,5 / 14,37
CMS	≅0 / 14,14	4,5 / 14,16	6,9 / 14,17	7,2 / 15
CES	≅0 / 14,12	2,9 / 14,26	3,5 / 14,44	6,5 / 14,86
Total	≅0 / 42,53	10,6 / 42,77	13,8 / 42,97	17,2 / 44,23

Tabela 7.1: Consumo de Processamento (%) / Memória (MB) do MHNCS.

Ao analisar os resultados obtidos, podemos observar, a partir da instanciação do primeiro provedor de contexto (Cenário 1), que o consumo do processador sobe para 10,6% enquanto o consumo de RAM aumenta apenas 0,24 MB. A partir daí, com a instanciação do segundo e terceiro provedores de contexto, o aumento médio nos consumos totais de processamento e memória são apenas, respectivamente, 3,3% e 0,7 3MB; já o aumento médio no uso dos recursos de processamento e memória, considerando-se os 4 cenários são, respectivamente, 5,73% e 0,56 MB. Deve ser observado que, embora o aumento médio no consumo do processador sendo considerados todos os cenários seja maior do que se considerados apenas os três últimos, o aumento médio no consumo de memória é levemente menor. Com esses resultados, é possível concluir que o MHNCS consome poucos recursos em dispositivos móveis compatíveis com os requisitos do projeto MobileHealthNet.

7.2 Avaliação Qualitativa

A avaliação qualitativa baseou-se nos resultados de um questionário e consiste num sistema de respostas de múltipla escolha, elaborado com base no conjunto de métricas definidas pela norma de qualidade de software ISO/IEC 9126 (NBR 13596) [1]. Esta norma lista o conjunto de características que devem ser verificadas em um software para que ele seja considerado um "software de qualidade". A norma NBR ISO/IEC 9126 abrange seis grandes grupos de características (e.g.

[3]http://developer.android.com/tools/debugging/ddms.html

Funcionalidade, Desempenho, Usabilidade, Manutenibilidade, Confiabilidade e Portabilidade).

Nesta análise qualitativa foram utilizadas as métricas usabilidade e manutenibilidade da norma NBR ISO/IEC 9126, com o objetivo de identificar os requisitos de qualidade que geraram resultados positivos para os desenvolvedores e não apenas medir o grau de satisfação deles. Para o papel de avaliador, foram selecionados alunos dos seguintes programas de: Programa de Pós-Graduação em Engenharia Elétrica (PPGEE) e Programa de Pós-Graduação em Ciência da Computação (PPGCC) ambos da Universidade Federal do Maranhão (UFMA). Esses alunos desenvolveram um projeto de implementação utilizando o *middleware* como parte do processo de avaliação da disciplina Computação Móvel. A descrição do cenário por eles implementado encontra-se no Apêndice C. Nesse caso, o sujeito considerado apto a avaliar o MHNCS não é um médico (usuários da aplicação), mas sim o desenvolvedor (usuário do *middleware*).

Na avaliação qualitativa, nos preocupamos em capturar as impressões e sugestões dos estudantes que utilizaram o MHNCS. Essa abordagem, chamada de validação pelo sujeito, está focada em descobrir se o MHNCS resolve de fato o problema para o qual foi concebido. Não é nosso objetivo avaliar a aplicação ciente de contexto, mas sim o *middleware* empregado para desenvolvê-la. Foi solicitado ao desenvolvedor que testasse os mecanismos de especificação, solicitação, obtenção, validação e distribuição de dados de contexto.

O questionário aplicado na avaliação é composto de quatro questões e as possíveis resposta para cada questão pode ser: "Não", "Parcialmente"ou "Sim". No Apêndice D é exposto o questionário para a verificação da usabilidade e manutenibilidade da infraestrutura proposta. O questionário aplicado na avaliação possui as seguintes questões:

1. O *middleware* disponibilizou um conjunto adequado de funcionalidades para facilitar o desenvolvimento de aplicações cientes de contexto?

2. As assinaturas dos métodos que compõem a interface do *middleware* são claras e fáceis de utilizar?

3. O *middleware* funcionou corretamente, conforme o esperado, sem apresentar falhas ou defeitos?

4. O software possibilitou o diagnóstico de eventuais erros da aplicação ocorridos durante o uso do *middleware* através de mensagens de retorno claras e oportunas?

De acordo a Tabela 7.2, podemos observar que a conclusão a partir das respostas obtidas pelos avaliadores participantes sobre o *middleware* proposto apresentou uma pontuação positiva de 100% com relação às funcionalidades disponíveis pela infraestrutura, sendo que 62,5% declararam que o conjunto de funcionalidades atende plenamente às expectativas dos desenvolvedores e 37,5% que as atende parcialmente. Destaca-se ainda que neste quesito não houveram proposições de novas funcionalidades e a reclamação mais usual refere-se a pouca documentação atualmente disponível. Já com relação à clareza e facilidade de uso das assinaturas dos métodos disponíveis pelas interfaces do MHNCS, este obteve uma pontuação positiva de 100%. Isso demostra que o *middleware* disponibiliza uma interface de programação adequada.

Questão	Não	Parcialmente	Sim
1	0%	37,5%	62,5%
2	0%	0,0%	100%
3	12,5%	62,5%	25,0%
4	0%	62,5%	37,5%

Tabela 7.2: Resultado da avaliação qualitativa

No que concerne à manutenibilidade, a grande maioria dos avaliadores considerou que alguns erros ocorridos apresentaram mensagens de exceções com fácil entendimento, possibilitando uma redução no tempo de sua correção durante o desenvolvimento. No entanto, em certos momentos eles sentiram dificuldade para corrigir eventuais erros, pois algumas mensagens não foram consideradas muito claras. Para minimizar esse problema, os avaliadores sugeriram uma padronização nas exceções com mensagens mais claras e oportunas com o intuito de facilitar a utilização do *middleware* proposto, o que está sendo levado em consideração em uma refatoração da API do mesmo.

As propriedades qualitativas que fazem parte dessa analise mostrou que nossa proposta proporciona uma facilidade no desenvolvimento de aplicações ciente de contexto, proporcionado por um conjunto de interfaces bem definidas. A agilidade de desenvolvimento pode ser decorrente de um infraestrutura que proporciona uma comunicação distribuída com acoplamento fraco entre as aplicações.

7.3 Considerações Finais

Do ponto de vista qualitativo, um conjunto de características do MHNCS ligadas à qualidade do software (manutenibilidade e usabilidade) foi positivo. Em resposta à avaliação global da infraestrutura e considerando todos os parâmetros analisados, todos os avaliadores responderam que é satisfatório o desempenho do *middleware*.

Também destacamos que algumas mensagens de erros não apresentaram uma fácil compreensão por parte dos avaliadores, mas não impossibilitaram que os desenvolvedores diagnosticassem os erros que ocorreram no desenvolvimento da aplicação. Do ponto de vista quantitativo, o *middleware* apresenta baixo consumo de memória e processamento, provando ser compatível com as especificações dos dispositivos mais acessíveis a população carente, público alvo do projeto MobileHealthNet.

Como sugestão dos avaliadores foi citado que a falta de uma boa documentação do *middleware*, acabou contribuindo para algumas dúvidas que foram surgindo no momento do desenvolvimento das aplicações. Assim, analisando as respostas e as considerações obtidas pelos questionários, conclui-se que o MHNCS obteve uma aprovação dos avaliadores.

8 CONCLUSÕES e TRABALHOS FUTUROS

Muitos dos avanços tecnológicos recentes vêm tornando a computação ubíqua uma realidade nas atividades diárias das pessoas. Entretanto, é essencial reduzir o esforço no desenvolvimento de aplicações ubíquas para facilitar o uso de todo o potencial deste paradigma. Aplicações para ambientes ubíquos devem atender a um conjunto de requisitos que impõem novos desafios aos desenvolvedores. Por exemplo, a sensibilidade ao contexto é um dos requisitos básicos da computação ubíqua, que envolve o uso de informações de contexto disponibilizadas por fontes distribuídas e heterogêneas espalhadas por esses ambientes. Aplicações precisam acessar essas informações para satisfazer os objetivos dos usuários. A especificação, solicitação, obtenção, análise e distribuição de dados de contexto é tipicamente realizada por plataformas de *middleware* de provisão e distribuição de contexto, que tornam menos complexo o processo de desenvolvimento das aplicações ciente de contexto.

A comunicação baseada no modelo *Publish/Subscribe* é frequentemente utilizada em aplicações ubíquas para difundir assincronamente as informações de contexto envolvidas nas constantes mudanças em seu estado de execução, geradas pelos ambientes altamente dinâmicos em que executam. Desse modo, as informações de contexto são disseminadas seletivamente, permitindo que componentes determinem seus interesses e somente recebam informações de acordo com esses interesses.

A utilização dos conceitos da computação ubíqua na área da saúde tem um grande impacto na forma comumente adotada no acompanhamento de pacientes. Assim, nossa contribuição foi implementar uma infraestrutura que facilita o desenvolvimento de aplicações relevantes para área da saúde, relacionado ao acompanhamento dos pacientes além dos encontros presenciais, utilizando os recursos da computação ubíqua que demonstrou ser um importante complemento no processo tradicional de acompanhamento dos pacientes.

Nesta dissertação, desenvolvemos a infraestrutura de *middleware* MHNCS que facilite a especificação, solicitação, obtenção, análise e distribuição de dados de contexto por parte das aplicações ciente de contexto a serem incorporadas ao projeto MobileHealthNet, cujo objetivo é fornecer o suporte ao desenvolvimento de aplicações cientes de contexto voltadas a área da saúde. Também avaliamos os mecanismos de distribuição das informações de contexto de outras plataformas de *middleware* relacionadas a este trabalho [17] [32] [52] [26] [34], destacados na literatura, assim como seus modelos de representação de dados de contexto, realizando um comparativo entre estes com este projeto.

O MHNCS é uma plataforma de integração que provê um modelo de representação de contexto orientado a objetos para facilitar a representação dos dados de contexto. O *middleware* integra mecanismo de distribuição de dados de contexto por diferentes fontes, utilizando ainda o modelo *Data-Centric Publish Subscribe* DCPS baseado na especificação DDS, garantindo um fraco acoplamento entre as aplicações clientes. A especificação possui diversas peculiaridades. Dentre elas, ser totalmente distribuída, disponibilizar políticas de QoS as quais foram utilizadas no trabalho para garantir a qualidade das informações que são definidas pelas aplicações.

Uma outra contribuição foi o projeto e implementação de mecanismos de notificação baseado em eventos. Desse modo, a informação de contexto é distribuída seletivamente, permitindo que as aplicações determinem seus interesses e somente recebam informações de acordo com esses. Mecanismos de comunicação baseados em eventos disponibilizam um suporte para composição de regras que permite uma maior expressividade na declaração de interesses. Portanto, esta dissertação apresentou uma solução que facilita o desenvolvimento aplicações ciente de contexto, disponibilizando mecanismo de distribuição centrado em dados de contexto com qualidade e um mecanismo de notificação baseado em eventos.

8.1 Principais Contribuições

Este trabalho teve como produto as seguintes contribuições:

- Definição de uma arquitetura com suporte à computação sensível ao contexto centrado em dados para ser integrada ao *middleware* MobileHealthNet;

- Implementação da infraestrutura de distribuição das informações baseado no modelo *Data-Centric Publish Subscribe* que possibilita um fraco acoplamento entre produtores e consumidores;

- Elaboração de uma linguagem baseado em XML para facilitar a definição de requisitos de dados de contexto, possibilitando que sejam incluídos parâmetros de QoC neste processo;

- Implementação do gerenciador de notificações baseados em eventos.

Como parte da divulgação dos resultados alcançados foi publicado os seguintes artigos científicos:

- A. Teles, D. Pinheiro, J. Gonçalves, R. Batista, F. Silva, V. Pinheiro, E. Haeusler, and M. Endler. **Mobilehealthnet: A middleware for mobile social networks in m-health**. In Proceedings of the 3rd International Conference on Wireless Mobile Communication and Healthcare, MobiHealth '12, 2012.

- A. S. Teles, D. Pinheiro, J. Gonçalves, R. Batista, V. Pinheiro, F. J. da Silva e Silva, and M. Endler. **Redes sociais móveis: Conceitos, aplicações e aspectos de segurança e privacidade**. In SBRC '13: Anais dos Minicursos do Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, pages 51–100. SBC, 2013.

- D.N. Pinheiro, A.S. Teles, B.T.P. Gomes, F.J. Silva e Silva. **A Middleware for Developing Context-aware Mobile Applications in Low-cost Acquisition Devices**. in Proceedings of the 7th FTRA International Conference on Human-centric Ubiquitous Computing and Applications (HumanCom-14). Ostrava, Czech Republic, August, 2014. Lecture Notes in Electrical Engineering (LNEE), Springer.

8.2 Trabalhos Futuros

Ao longo do desenvolvimento deste trabalho observamos diversas oportunidades para o desenvolvimento de pesquisas que complementem as já aqui

realizadas. Uma delas, seria o desenvolvimento de um serviço de persistência dos dados de contexto. Com isso, informações de contexto poderiam ser armazenadas por longos prazos de tempo, permitindo que as aplicações utilizem dados anteriores para melhorar seus mecanismo de inferência.

Visando melhorar o mecanismo de distribuição dos dados de contexto é possível implementar ainda outros conceitos de QoC no *middleware* para manter a qualidade das informações em condições adversas (falhas na comunicação e diminuição da largura de banda), bem como a implementação de novos componentes para inferir a situação do paciente a partir dos dados de contexto obtidos em cenários de realização de atividades físicas.

Finalmente, uma outra possível linha de pesquisa seria o desenvolvimento de mecanismos de renegociação em tempo de execução dos parâmetros de QoC especificados pela aplicação, bem como a adaptação dinâmica de componentes do *middleware* para melhor contornar variações do ambiente de execução.

Referências Bibliográficas

[1] *Associação Brasileira de Normas Técnicas. ABNT NBR ISO/IEC 91261: Engenharia de Software - qualidade de produto - parte 1 : modelo de qualidade*. Associação Brasileira de Normas Técnicas, 2003.

[2] W. F. Ableson, C. King, and R. Sen. *Android em Ação, 3ª Ed*. Elsevier Brasil, 2012.

[3] B. Adams, D. Phung, and S. Venkatesh. Sensing and using social context. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, 5(2):11, 2008.

[4] V. P. Almeida, M. Endler, and E. H. Haeusler. Patient-buddy-build: Customized mobile monitoring for patients with chronicle diseases. In *Proceedings of the 5th International Conference on eHealth, Telemedicine, and Social Medicine*, eTELEMED '13, Nice, France, 2013.

[5] M. Baldauf, S. Dustdar, and F. Rosenberg. A survey on context-aware systems. *Int. J. Ad Hoc Ubiquitous Comput.*, 2(4):263–277, June 2007.

[6] M. Barua, M. Shamsul, A. Sarker, X. Liang, and X. Shen. Secure and quality of service assurance scheduling scheme for wban with application to ehealth. In *Wireless Communications and Networking Conference (WCNC), 2011 IEEE*, pages 1102–1106, March 2011.

[7] R. Batista and F. J. d. S. e. Silva. Uma infraestrutura de comunicação para colaboração em redes sociais móveis. In *Simpósio Brasileiro de Sistemas Colaborativos*, SBSC '12, 2012.

[8] P. Bellavista, A. Corradi, M. Fanelli, and L. Foschini. A survey of context data distribution for mobile ubiquitous systems. *ACM Comput. Surv.*, 44(4):24:1–24:45, Sept. 2012.

[9] C. Bolchini, C. A. Curino, G. Orsi, E. Quintarelli, R. Rossato, F. A. Schreiber, and L. Tanca. And what can context do for data? *Commun. ACM*, 52(11):136–140, Nov. 2009.

[10] C. Borcea, A. Gupta, A. Kalra, Q. Jones, and L. Iftode. The mobisoc middleware for mobile social computing: Challenges, design, and early experiences. In *Proceedings of the 1st International Conference on MOBILe Wireless MiddleWARE, Operating Systems, and Applications*, MOBILWARE '08, pages 27:1–27:8, ICST, Brussels, Belgium, Belgium, 2007. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[11] D. M. Boyd and N. B. Ellison. Social network sites: Definition, history, and scholarship. *Journal of Computer-Mediated Communication*, 13(1):210–230, 2007.

[12] T. Buchholz, A. Küper, and M. Schiffers. Quality of context: What it is and why we need it. In *In Proceedings of the 10th Workshop of the OpenView University Association: (HPOVUA)*, 2003.

[13] G.-P. Castellote. Omg data-distribution service: architectural overview. In *Distributed Computing Systems Workshops, 2003. Proceedings. 23rd International Conference on*, pages 200–206, May 2003.

[14] G. Chen and D. Kotz. Solar: A pervasive-computing infrastructure for context-aware mobile applications. Technical Report TR2002-421, Dartmouth College, 2002.

[15] M. Chen, S. Gonzalez, A. Vasilakos, H. Cao, and V. C. Leung. Body area networks: A survey. *Mobile Networks and Applications*, 16(2):171–193, 2011.

[16] S. Chetan, A. Ranganathan, and R. Campbell. Towards fault tolerance pervasive computing. *Technology and Society Magazine, IEEE*, 24(1):38–44, 2005.

[17] A. Copetti, O. Loques, J. C. Leite, T. P. Barbosa, and A. C. Nobrega. Intelligent context-aware monitoring of hypertensive patients. In *Pervasive Computing Technologies for Healthcare, 2009. PervasiveHealth 2009. 3rd International Conference on*, pages 1–6. IEEE, 2009.

[18] L. David, R. Vasconcelos, L. Alves, R. André, G. Baptista, and M. Endler. A communication middleware for scalable real-time mobile collaboration. *IEEE 21st International WETICE Conference, Track on Adaptive and Reconfigurable Service-oriented and component-based Applications and Architectures (AROSA)*, june 2012.

[19] A. K. Dey. *Providing Architectural Support for Building Context-aware Applications*. PhD thesis, Atlanta, GA, USA, 2000. AAI9994400.

[20] A. K. Dey. Understanding and using context. *Personal Ubiquitous Comput.*, 5(1):4–7, Jan. 2001.

[21] C. Emmanouilidis, R.-A. Koutsiamanis, and A. Tasidou. Mobile guides: Taxonomy of architectures, context awareness, technologies and applications. *Journal of Network and Computer Applications*, 36(1):103–125, 2013.

[22] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys (CSUR)*, 35(2):114–131, June 2003.

[23] A. Gaddah and T. Kunz. A survey of middleware paradigms for mobile computing. Technical report, Carleton University, 2003.

[24] J. Gonçalves, A. Teles, and F. J. d. S. Silva. Um modelo de segurança e privacidade para redes sociais móveis aplicadas à área da saúde. In *XII Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*, SBSeg '12, 2012.

[25] T. Gu, H. K. Pung, and D. Q. Zhang. Toward an osgi-based infrastructure for context-aware applications. *Pervasive Computing, IEEE*, 3(4):66–74, Oct 2004.

[26] A. Gupta, A. Kalra, D. Boston, and C. Borcea. Mobisoc: a middleware for mobile social computing applications. *Mob. Netw. Appl.*, 14:35–52, February 2009.

[27] A. Held, S. Buchholz, and A. Schill. Modeling of context information for pervasive computing applications. In *Procceding of the World Multiconference on Systemics, Cybernetics and Informatics*. Springer, 2002.

[28] K. Henricksen, J. Indulska, and A. Rakotonirainy. Modeling context information in pervasive computing systems. In *Proceedings of the First International Conference on Pervasive Computing*, Pervasive '02, pages 167–180, London, UK, UK, 2002. Springer-Verlag.

[29] Q. Huang and Y. Liu. On geo-social network services. In *Geoinformatics, 2009 17th International Conference on*, pages 1–6. IEEE, 2009.

[30] R. Joshi and G.-P. Castellote. A comparison and mapping of data distribution service and high-level architecture. Technical report, RTI, Inc, 2004.

[31] N. Kayastha, D. Niyato, P. Wang, and E. Hossain. Applications, architectures, and protocol design issues for mobile social networks: A survey. *Proceedings of the IEEE*, 99(12):2130–2158, Dec 2011.

[32] Y. Lee, S. S. Iyengar, C. Min, Y. Ju, S. Kang, T. Park, J. Lee, Y. Rhee, and J. Song. Mobicon: A mobile context-monitoring platform. *Commun. ACM*, 55(3):54–65, Mar. 2012.

[33] J. Li and Q. Li. Decentralized self-management of trust for mobile ad hoc social networks. *International Journal of Computer Networks & Communications (IJCNC)*, 3(6):1–17, 2011.

[34] F. Lopes, F. C. Delicato, T. Batista, E. Cavalcante, T. Pereira, P. F. Pires, P. Ferreira, and R. Mendes. Opencopi: Middleware integration for ubiquitous computing. *Int. J. Parallel Emerg. Distrib. Syst.*, 29(2):178–212, Mar. 2014.

[35] K. C. Louden. *Compiladores - Princípios e Práticas*. Pioneira Thomson Learning, 2004.

[36] X. Lu, X. Li, T. Yang, Z. Liao, W. Liu, and H. Wang. Qos-aware publish-subscribe service for real-time data acquisition. In M. Castellanos, U. Dayal, and T. Sellis, editors, *Business Intelligence for the Real-Time Enterprise*, volume 27 of *Lecture Notes in Business Information Processing*, pages 29–44. Springer Berlin Heidelberg, 2009.

[37] R. Lübke. *Ein Framework zur Entwicklung mobiler Social Software auf Basis von Android*. PhD thesis, Dresden, Germany, March 2011.

[38] R. Lübke, D. Schuster, and A. Schill. Mobilisgroups: Location-based group formation in mobile social networks. In *Proceedings of the 9th Annual IEEE International Conference on Pervasive Computing and Communications, Workshop Proceedings*, PerCom '11, pages 502–507, Seattle, WA, USA, 2011. IEEE.

[39] M. Malcher, J. Aquino, H. Fonseca, L. David, A. Valeriano, and M. Endler. A middleware supporting adaptive and location-aware mobile collaboration. In *Mobile Context Workshop: Capabilities, Challenges and Applications, Adjunct Proceedings of UbiComp*, 2010.

[40] A. Manzoor, H.-L. Truong, and S. Dustdar. Quality of context: Models and applications for context-aware systems in pervasive environments. *The Knowledge*

Engineering Review, Special Issue on Web and Mobile Information Services, 29(02):154–170, 2011.

[41] J. McCarthy. Notes on formalizing context. In *Proceedings of the 13th International Joint Conference on Artifical Intelligence - Volume 1*, IJCAI'93, pages 555–560, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

[42] M. Moeiz, T. Chakib, F. Jaouhar, and A. C. Ben. Dynamic context-aware and limited resources-aware service adaptation for pervasive computing. *Advances in Software Engineering*, 2011:7:7–7:7, Jan. 2011.

[43] D. R. Morse, S. Armstrong, and A. K. Dey. The what, who, where, when, why and how of context-awareness. In *CHI '00 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '00, pages 371–371, New York, NY, USA, 2000. ACM.

[44] P. Ötürk and A. Aamodt. Towards a model of context for case-based diagnostic problem solving. In *Proceedings of the First International and Interdisciplinary Conference on Modeling and Using Context*, Lecture Notes in Computer Science, pages 198–208. Springer, 1997.

[45] R. M. Pessoa. Infraware: Um Middleware de Suporte a Aplicações Sensíveis ao Contexto. Master's thesis, Universidade Federal do Espírito Santo (UFES), Vitória, ES, Brazil, 2006.

[46] S. Poslad. *Ubiquitous Computing: Smart Devices, Environments and Interactions*. Wiley Publishing, 1st edition, 2009.

[47] V. Sacramento, M. Endler, H. K. Rubinsztejn, L. d. S. Lima, K. Gonçalves, and G. A. Bueno. An architecture supporting the development of collaborative applications for mobile users. In *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2004. WET ICE 2004. 13th IEEE International Workshops on*, pages 109–114. IEEE, 2004.

[48] P. Saint-Andre, K. Smith, and R. Tronçon. *XMPP: The Definitive Guide Building Real-Time Applications with Jabber Technologies*. O'Reilly Media, Inc., 2009.

[49] B. Schilit, N. Adams, and R. Want. Context-aware computing applications. In *Proceedings of the 1994 First Workshop on Mobile Computing Systems and Applications*, WMCSA '94, pages 85–90, Washington, DC, USA, 1994. IEEE Computer Society.

[50] B. N. Schilit and M. M. Theimer. Disseminating active map information to mobile hosts. *Network, IEEE*, 8(5):22–32, Sept. 1994.

[51] D. C. Schmidt and H. v. Hag. Addressing the challenges of mission-critical information management in next-generation net-centric pub/sub systems with opensplice dds. pages 1 –8, april 2008.

[52] D. Schuster, I. Koren, T. Springer, D. Hering, B. Söllner, M. Endler, and A. Schill. *Creating Applications for Real-Time Collaboration with XMPP and Android on Mobile Devices*. Handbook of Research on Mobile Software Engineering: Design, Implementation and Emergent Applications, IGI Global, 2012.

[53] D. Schuster, A. Rosi, M. Mamei, T. Springer, M. Endler, and F. Zambonelli. Pervasive social context: Taxonomy and survey. *ACM Transactions on Intelligent Systems and Technology*, 4(3):46:1–46:22, July 2013.

[54] P. R. d. S. Soares. Desenvolvimento de Propaganda Interativa e Sensível ao Contexto para TV Digital, 2010. Monografia (Graduação em Ciência da Computação), UFPE (Universidade Federal de Pernambuco), Brazil.

[55] B. Sollner. *XMPP-based Media Sharing for Mobile Collaboration with Android Phones*. PhD thesis, Technische Universitat Dresden, Germany, Octuber 2009.

[56] S. Stan. Data-centric pervasive information is the wave of the future. *Rf design*, aug 2006.

[57] T. Strang and C. Linnhoff-Popien. A context modeling survey. In *Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing, Nottingham/England*, 2004.

[58] A. Teles, J. Gonçalves, V. Pinheiro, F. J. d. S. Silva, and M. Endler. Infraestrutura e aplicações de redes sociais móveis para colaboração em saúde. In *Congresso Brasileiro de Informática em Saúde*, CBIS '12, 2012.

[59] A. Teles, D. Pinheiro, J. Gonçalves, B. Rômulo, F. Silva, V. Pinheiro, E. Haeuslerm, and M. Endler. Mobilehealthnet: A middleware for mobile social networks in m-health. In *Proceedings of the 3rd International Conference on Wireless Mobile Communication and Healthcare*, MobiHealth '12, 2012.

[60] A. Teles, D. Pinheiro, J. Gonçalves, R. Batista, V. Pinheiro, F. J. d. S. Silva, and M. Endler. Redes sociais móveis: Conceitos, aplicações e aspectos de segurança e privacidade. In *SBRC '13: Anais dos Minicursos do Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 51–100. SBC, 2013.

[61] A. Teles, F. J. d. S. Silva, and R. Batista. *Security and Privacy in Mobile Social Networks*. Lecture Notes in Social Networks. Security and Privacy Preserving in Social Networks, Springer, 2013.

[62] J. M. L. Vega. Plataforma de Trabajo Colaborativo sobre Middleware DDS. Master's thesis, Universidad de Granada, Spain, 2008.

[63] C. Vicente, D. Freni, C. Bettini, and C. Jensen. Location-related privacy in geo-social networks. *Internet Computing, IEEE*, 15:20–27, may-june 2011.

[64] V. Vieira, D. Souza, A. C. Salgado, and P. Tedesco. Uso e representação de contexto em sistemas computacionais. In *Tópicos em Sistemas Interativos e Colaborativos*, pages 127–166, São Carlos, São Paulo, Brazil, 2006. UFSCAR.

[65] V. Vieira, P. Tedesco, and A. C. Salgado. A process for the design of context-sensitive systems. In *Proceedings of the 2009 13th International Conference on Computer Supported Cooperative Work in Design*, CSCWD '09, pages 143–148, Washington, DC, USA, 2009. IEEE Computer Society.

[66] V. Vieira, P. Tedesco, and A. C. Salgado. Designing context-sensitive systems: An integrated approach. *Expert Systems with Applications*, 38(2):1119–1138, Feb. 2011. Intelligent Collaboration and Design.

[67] N. Wang, D. C. Schmidt, H. v. Hag, and A. Corsaro. Toward an adaptive data distribution service for dynamic large-scale network-centric operation and warfare (ncow) systems. In *Military Communications Conference, 2008. MILCOM 2008. IEEE*, pages 1 –7, nov. 2008.

[68] S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications*. Structural Analysis in the Social Sciences. Cambridge University Press, 1 edition, 1994.

[69] J. G. Webster. *The measurement, instrumentation, and sensors handbook*. Berlin: Springer, 1999.

[70] M. Weiser. The computer for the twenty-first century. *SIGMOBILE Mobile Computing and Communications Review.*, 9:94–100, Sept. 1991.

[71] M. Xiong, J. Parsons, J. Edmondson, H. Nguyen, and D. C. Shmidt. Evaluating the performance of publish/subscribe platforms for information management in distributed real-time and embedded systems. Technical report, 2010.

Apêndices

A Estrutura da Linguagem de Descrição de Requisitos de Contexto

Neste apêndice, apresentamos as definições de esquema XML da linguagem de especificação de requisitos de contexto que foi proposto na dissertação. Essa estrutura expressa um conjunto de regras a que um documento XML deve estar de acordo para ser considerado "válido"de acordo com esse esquema.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
3
4    <xs:element name="parameter">
5       <xs:complexType>
6        <xs:attribute name="name" use="required" type="xs:string"/>
7        <xs:attribute name="value" use="required" type="xs:string"/>
8       </xs:complexType>
9    </xs:element>
10
11   <xs:element name="configuration">
12     <xs:complexType>
13       <xs:sequence maxOccurs="unbounded" minOccurs="0">
14         <xs:element ref="parameter" />
15       </xs:sequence>
16     </xs:complexType>
17   </xs:element>
18
19   <xs:attribute name="name">
20         <xs:simpleType>
21       <xs:restriction base="xs:string">
22         <xs:enumeration value="ACCURACY"/>
23         <xs:enumeration value="FREQUENCY"/>
24         <xs:enumeration value="FRESHNESS"/>
25         <xs:enumeration value="REFRESH_RATE"/>
26       </xs:restriction>
27     </xs:simpleType>
28     </xs:attribute>
```

```
29
30    <xs:attribute name="kind">
31        <xs:simpleType>
32      <xs:restriction base="xs:string">
33        <xs:pattern id="FREQUENCY" value="EXACT|MINIMUM|MAXIMUM"/>
34        <xs:pattern id="REFRESH_RATE" value="GET_ALL|GET_LAST"/>
35      </xs:restriction>
36    </xs:simpleType>
37    </xs:attribute>
38
39    <xs:attribute name="op">
40        <xs:simpleType>
41      <xs:restriction base="xs:string">
42        <xs:enumeration id="equal" value="eq" />
43        <xs:enumeration id="greater_than" value="gt" />
44        <xs:enumeration id="less_than" value="lt" />
45        <xs:enumeration id="greater_than_or_equal" value="gte" />
46        <xs:enumeration id="less_than_or_equal" value="lte" />
47      </xs:restriction>
48    </xs:simpleType>
49    </xs:attribute>
50
51  <xs:element name="property">
52     <xs:complexType>
53      <xs:attribute ref="name" use="required"/>
54      <xs:attribute name="time" use="optional" type="xs:int"/>
55      <xs:attribute ref="kind" use="optional"/>
56      <xs:attribute name="age" use="optional" type="xs:int"/>
57      <xs:attribute ref="op" use="optional"/>
58      <xs:attribute name="amount" use="optional" type="xs:int"/>
59     </xs:complexType>
60  </xs:element>
61
62  <xs:element name="qualityOfContext">
63    <xs:complexType>
64      <xs:sequence maxOccurs="unbounded" minOccurs="0">
65        <xs:element ref="property" />
66      </xs:sequence>
67    </xs:complexType>
68  </xs:element>
```

```
69
70    <xs:attribute name="init" default="SINGLE">
71          <xs:simpleType>
72        <xs:restriction base="xs:string">
73          <xs:enumeration id="single" value="SINGLE" />
74          <xs:enumeration id="all" value="ALL" />
75        </xs:restriction>
76      </xs:simpleType>
77      </xs:attribute>
78
79    <xs:element name="information">
80      <xs:complexType>
81        <xs:attribute name="id" type="xs:string" use="required"/>
82        <xs:attribute ref="init" use="required"/>
83      </xs:complexType>
84    </xs:element>
85
86    <xs:element name="contextInformation">
87      <xs:complexType>
88        <xs:sequence maxOccurs="unbounded" minOccurs="1">
89          <xs:element ref="information" />
90        </xs:sequence>
91      </xs:complexType>
92    </xs:element>
93
94    <xs:element name="requirement">
95      <xs:complexType>
96        <xs:sequence maxOccurs="1" minOccurs="1">
97          <xs:element ref="contextInformation" />
98          <xs:element ref="qualityOfContext"/>
99          <xs:element ref="configuration"/>
100       </xs:sequence>
101     </xs:complexType>
102   </xs:element>
103 </xs:schema>
```

Listagem A.1: XSD para definição de requisitos de contexto

B Estrutura da Linguagem de definição de Eventos

Neste apêndice, apresentamos as definições de esquema XML da linguagem de definição de eventos que foi proposto na dissertação. Essa estrutura expressa um conjunto de regras a que um documento XML deve estar de acordo para ser considerado "válido"de acordo com esse esquema.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
3    <xs:attribute name="operator" default="eq">
4      <xs:simpleType>
5        <xs:restriction base="xs:string">
6          <xs:enumeration id="equal" value="eq" />
7          <xs:enumeration id="greater_than" value="gt" />
8          <xs:enumeration id="less_than" value="lt" />
9          <xs:enumeration id="greater_than_or_equal" value="gte" />
10         <xs:enumeration id="less_than_or_equal" value="lte" />
11       </xs:restriction>
12     </xs:simpleType>
13   </xs:attribute>
14   <xs:attribute name="connector" default="none">
15     <xs:simpleType>
16       <xs:restriction base="xs:string">
17         <xs:enumeration value="none" />
18         <xs:enumeration value="and" />
19         <xs:enumeration value="or" />
20       </xs:restriction>
21     </xs:simpleType>
22   </xs:attribute>
23   <xs:element name="condition">
24     <xs:complexType>
25       <xs:attribute name="context" use="required" />
26       <xs:attribute ref="operator" use="required" />
27       <xs:attribute name="value" use="required" />
28       <xs:attribute name="dataTypeValue" use="required" />
```

```
29        <xs:attribute ref="connector" use="optional" />
30      </xs:complexType>
31    </xs:element>
32    <xs:element name="expression">
33      <xs:complexType>
34        <xs:sequence maxOccurs="unbounded" minOccurs="1">
35          <xs:element ref="condition" />
36        </xs:sequence>
37      </xs:complexType>
38    </xs:element>
39    <xs:element name="rule">
40      <xs:complexType>
41        <xs:sequence maxOccurs="1" minOccurs="1">
42          <xs:element ref="expression" />
43        </xs:sequence>
44        <xs:attribute name="name" type="xs:string" />
45      </xs:complexType>
46    </xs:element>
47    <xs:element name="event">
48      <xs:complexType>
49        <xs:sequence maxOccurs="unbounded" minOccurs="1">
50          <xs:element ref="rule" />
51        </xs:sequence>
52      </xs:complexType>
53    </xs:element>
54    <xs:element name="requirement">
55      <xs:complexType>
56        <xs:sequence maxOccurs="1" minOccurs="1">
57          <xs:element ref="event" />
58        </xs:sequence>
59      </xs:complexType>
60    </xs:element>
61  </xs:schema>
```

Listagem B.1: XSD para definição de Evento

C Descrição do Cenário de Implementação

Uma vítima de violência pode recorrer a um sistema de atendimento a emergências policiais discando o número 190. A polícia militar possui diversos veículos (patrulhas) para o atendimento destas ocorrências. Utilizando os recursos do *middleware* MobileHealthNet, implemente um sistema que permita:

1. O acompanhamento das unidades móveis em campo por uma central de atendimento, visualizando-se em um mapa o deslocamento em tempo real destas unidades
2. O acompanhamento pela central de atendimento de quais unidades móveis estão livres e quais estão realizando atendimento
3. A comunicação entre a central de atendimento e as unidades móveis que permita a designação de uma unidade móvel para o atendimento de uma ocorrência
4. O registro textual e fotográfico/filmado das ocorrências atendidas

Para tanto, devem ser desenvolvidas as seguintes aplicações:

- Equipe 1: Uma aplicação a ser executada em um computador da central de atendimento da polícia militar para monitoramento do deslocamento das viaturas policiais. Indique através de um padrão de cores quais estão livres e quais estão ocupadas. Ao selecionar uma viatura livre, disponibilize duas opções: designar ocorrência e chat. Caso seja selecionada a opção designar ocorrência, abra uma janela para a informar uma descrição da ocorrência e o endereço da mesma. Caso seja selecionado o *chat*, abra uma tela para a realização de *chat* com a viatura;
- Equipe 2: Uma aplicação a ser executada na viatura policial. Esta aplicação deve notificar solicitações para atendimento a ocorrências (vindas da central de atendimento). Os dados da ocorrência devem ser mostrados e o policial deve indiciar que visualizou a ocorrência a ele designada, mudando seu estado para

ocupado. Durante o atendimento à ocorrência o policial deve poder enviar um relatório contendo um registro textual e fotográfico ou filmado da ocorrência atendida. Ao enviar o relatório seu estado muda novamente para livre;

- Equipe 3: Um simulador de deslocamento a ser executado pelo emulador da viatura policial. Este simulador inicia escolhendo um ponto aleatório de um espaço geográfico e, a partir daí, faz um deslocamento seguindo o padrão *random waypoint* até o recebimento de uma ocorrência. Ao receber a designação da ocorrência a viatura se desloca até o ponto informado da mesma e nele pára até que término do atendimento (tornar-se livre novamente) quando, então, um novo deslocamento seguindo o padrão *random waypoint* é traçado. A equipe deve ainda implementar uma aplicação a ser executada em computadores da polícia militar que permita listar as ocorrências atendidas (ordenadas pelo tempo) e visualizar o relatório textual, fotográfico e filmado das mesmas.

D Questionário de Avaliação Qualitativo

Prezados desenvolvedores, o objetivo deste questionário é colher informações que nos permita avaliar a usabilidade e manutenibilidade do serviço de contexto do *middleware* MobileHealthNet. Pedimos que dedique um pouco do seu tempo para preenche-lo de forma criteriosa e construtiva. Suas informações serão muito importantes para o aprimoramento da ferramenta. Sugerimos que você complemente a resposta para cada questão com sugestões de melhorias ou quaisquer outros comentários que julgar pertinentes. Agradecemos a sua participação !

1) O middleware disponibilizou um conjunto adequado de funcionalidades para facilitar o desenvolvimento de aplicações cientes de contexto?.

   a) Não disponibilizou (são insuficientes).
   b) Disponibilizou parcialmente.
   c) Sim, disponibilizou completamente.

   Resposta: ( )

   _____

   _____

   _____

   _____

   _____

   _____

   _____

   _____

2) As assinaturas dos métodos que compõem a interface do middleware são claras e fácies de utilizar?.

   a) Não.

b) Parcialmente.

c) Sim.

Resposta: ( )

3) O middleware funcionou corretamente, conforme o esperado, sem apresentar falhas ou defeitos?

a) Não, apresentou falhas ou defeitos que foram de difícil compreensão.

b) Parcialmente, apresentou falhas ou defeitos porém de fácil compreensão.

c) Sim, não ocorreram falhas ou defeitos.

Resposta: ( )

4) O software possibilitou o diagnóstico de eventuais erros da aplicação ocorridos durante o uso do middleware através de mensagens de retorno claras e oportunas?

a) Não ocorreram mensagens de erro (minha aplicação sempre funcionou corretamente).

b) Parcialmente, apresentou mensagens de erros, mas não foram muito claras.

c) Sim, apresentou mensagens de erros claras e de fácil compreensão.

Resposta: ( )