UNIVERSIDADE CATÓLICA DE PELOTAS
CENTRO DE CIÊNCIAS SOCIAIS E TECNOLÓGICAS
MESTRADO EM ENGENHARIA ELETRÔNICA E COMPUTAÇÃO

DOUGLAS ADALBERTO SCHEUNEMANN

Ciência de Situação na IoT: Uma Arquitetura Explorando Processamento Híbrido de Contexto

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Engenharia Eletrônica e Computação

Orientador: Prof. Adenauer Correa Yamin

Pelotas
2016

CIP — CATALOGAÇÃO NA PUBLICAÇÃO

Scheunemann, Douglas Adalberto

Ciência de Situação na IoT:
Uma Arquitetura Explorando Processamento Híbrido de Contexto
/ Douglas Adalberto Scheunemann. – Pelotas: 2016.

81 f.: il.

Dissertação (mestrado) – Universidade Católica de Pelotas. 2016. Orientador: Adenauer Correa Yamin.

I. Yamin, Adenauer Correa. II. Título.

“Tente mover o mundo –

o primeiro passo será mover a si mesmo.” — PLATÃO

AGRADECIMENTOS

Ao final desta jornada, tenho a satisfação de agradecer algumas pessoas que foram fundamentais para o seu êxito.

Agradeço primeiramente à minha família, que me incentivou e deu suporte durante a realização do mestrado. Agradeço especialmente à minha esposa Grace, pela compreensão e apoio. Agradeço à minha filha Poliana, que chegou nos últimos dois meses do mestrado, e que representa uma nova fonte de motivação na minha vida.

Agradeço ao meu orientador Adenauer Yamin, um verdadeiro líder, com o dom de motivar as pessoas ao seu redor. Agradeço por seus ensinamentos e pela sua orientação, disponível em qualquer lugar, a qualquer momento e por qualquer canal de comunicação.

Agradeço também ao João Ladislau Lopes pelo apoio em diversas etapas e pelas discussões técnicas que contribuíram profundamente para o trabalho.

Aos meus amigos Juliano Rosinha Barboza e Gustavo Ott pela troca de experiências, ensinamentos e apoio.

Aos sócios da empresa Contronic, Maurício Tavares e Sidinei Seus pelo incentivo e compreensão nos momentos em que estive ausente.

Por fim, agradeço a todos os professores do Mestrado em Engenharia Eletrônica e Computação da UCPel pelo convívio e aprendizado.

RESUMO

A *Internet of Things* (IoT) vem influenciando a maneira como os sistemas computacionais são desenvolvidos, possibilitando uma interação mais proativa com os usuários, expandindo características de mobilidade e disponibilidade. Nesse cenário, cresce a demanda por aplicações que possam reconhecer o contexto do usuário e fornecer serviços baseados em sua situação. A identificação de situações representa um desafio de pesquisa para aplicações em IoT, dada a complexidade das relações que precisam ser estabelecidas e processadas até que se obtenha informações no nível de abstração suficiente para a identificação das situações de interesse das aplicações. Dados de diversas fontes podem ser utilizados durante o processamento contextual, o qual pode ocorrer em múltiplas etapas e envolver diferentes técnicas, baseadas em especificação, aprendizado ou em modelos híbridos onde ambas são combinadas. O uso de *middlewares* é destacado na literatura como uma forma de tratar a heterogeneidade de dispositivos na IoT e também para tornar o processamento contextual mais transparente para as aplicações. No entanto, para que um *middleware* seja independente do domínio de aplicação é necessário acrescentar em sua arquitetura camadas que permitam o gerenciamento dos componentes de software aplicados no processamento contextual, e que possibilitem ainda a composição de diferentes fluxos de processamento contextual. Considerando esta demanda, o objetivo desta dissertação é a concepção de uma arquitetura voltada para o gerenciamento e composição de fluxos de processamento contextuais híbridos para prover ciência de situação para aplicações em IoT. Uma das premissas da arquitetura é a sua integração com o *middleware* EXEHDA (*Execution Environment for Highly Distributed Applications*). A avaliação da arquitetura foi feita através de dois cenários de uso, um na área de reabilitação cardíaca e outro na área de gerenciamento de ambientes hospitalares. Os resultados obtidos se mostraram promissores, apontando para continuidade da pesquisa.

Palavras-chave: Ciência de Situação. Processamento Contextual. Modelos Híbridos. Internet das Coisas.

Situation Awareness in IoT:
An Architecture Exploring Hybrid Context Processing

ABSTRACT

The Internet of Things (IoT) has influenced the development of computational systems, enabling a more proactive interaction with users, expanding features as mobility and availability. In this scenario increases the demand for applications that can recognize the user's context and can provide situation based services. The identification of situations is a research challenge for applications in the IoT, given the complexity of the relationships that must be established and processed. Data from several sources can be used for contextual processing, which can occur in multiple steps and involve different techniques based on specification, learning or hybrid models in which both are combined. The use of middleware is highlighted in the literature as a way of treating heterogeneous devices in the IoT and also to make more transparent the contextual processing for the applications. However, for a middleware be independent of the application domain is necessary to add in its architecture layers that allow the management of software components used in contextual processing and still allow the composition of different contextual processing flows. Considering this demand, the aim of this work is to design an architecture for management and composition of hybrid contextual processing flows to provide situation awareness for IoT applications. One of the premises of the architecture is its integration with the middleware EXEHDA (Execution Environment for Highly Distributed Applications). The evaluation of the architecture was done through two scenarios of use, one in the area of cardiac rehabilitation and another in the area of management of hospital environments. The results obtained were promising, pointing to the continuity of the research.

Keywords: Situation Awareness, Contextual Processing, Hybrid Models, Internet of Things.

LISTA DE ABREVIATURAS E SIGLAS

AAL	*Ambient Assisted Living*
ANN	*Artificial Neural Network*
API	*Application Programming Interface*
ARFF	*Attribute-Relation File Format*
BSN	*Body Sensor Network*
CPDL	*Context Processing Definition Language*
EXEHDA	*Execution Environment for Highly Distributed Applications*
EXEHDA-IS	EXEHDA – *IoT Situations*
FFT	*Fast Fourier Transform*
GPIO	*General Purpose Input/Output*
HMM	*Hidden Markov Model*
HTTP	*Hypertext Transfer Protocol*
HUSFP	Hospital Universitário São Francisco de Paula
I2C	*Inter-Integrated Circuit*
IA	Inteligência Artificial
IoT	*Internet of Things*
JSON	*JavaScript Object Notation*
OWL	*Web Ontology Language*
REST	*Representational State Transfer*
RMI	*Remote Method Invocation*
SOA	*Service-Oriented Architecture*
SoC	*System on Chip*

SPI *Serial Peripheral Interface*

UART *Universal Asynchronous Receiver/Transmitter*

UCPel Universidade Católica de Pelotas

URI *Uniform Resource Identifier*

XML *Extensible Markup Language*

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Com os avanços ocorridos nas áreas de sensores, redes sem fio, dispositivos móveis e computação em nuvem, os conceitos de computação ubíqua propostos por Mark Weiser podem ser vistos atualmente em aplicações reais. Aplicações estas que agem de maneira proativa, identificando o contexto do usuário e fornecendo serviços otimizados considerando o mesmo, tornando a interação com os sistemas computacionais mais intuitiva e livre de distrações (PERERA et al., 2013).

Nesse cenário a IoT, utilizada como estratégia de ubiquidade, permite que uma grande variedade de dispositivos possa ser conectada através da infraestrutura disponibilizada pela internet. Segundo dados da ITU (*International Telecommunication Union*), em 2020 o número de dispositivos conectados poderá chegar a 40 bilhões.

A heterogeneidade dos elementos de hardware e software da IoT torna o uso de *middlewares* essencial para prover a interoperabilidade entre os dispositivos e facilitar o desenvolvimento de aplicações (RAZZAQUE et al., 2016). Um *middleware* é uma camada de software utilizada entre aplicações e os provedores de informações e serviços baseados em tecnologias de internet. O mesmo agrupa funções comuns entre aplicações, em áreas como: interoperabilidade, gerenciamento de dispositivos, segurança e ciência de situação (PERERA et al., 2013).

A ciência de situação representa a funcionalidade de um sistema computacional em obter uma visão abrangente e em alto nível de abstração dos contextos de interesse das aplicações, que pode ser utilizada em seu processo de tomada de decisão (PERERA et al., 2013; BIBRI, 2015). Segundo Perera et al. (2013) grande parte dos *middlewares* para IoT não possuem ainda funções para detecção de contexto e inferência de situação. Dentre as causas para esta ausência, pode ser destacada a complexidade das camadas de software que necessitam ser integradas aos *middlewares* para disponibilizar estas funções.

1.1 Tema de pesquisa

O tema deste trabalho é a concepção de uma arquitetura de software, integrada ao *middleware* EXEHDA (*Execution Environment for Highly Distributed Applications*) (LOPES et al., 2014a), nomeada EXEHDA-IS (EXEHDA – *IoT Situations*) voltada para o gerenciamento e composição de mecanismos do processamento contextual com o intuito de prover ciência de situação para aplicações em IoT.

1.2 Motivações e objetivos

Possibilitando o acesso e a interação com uma grande quantidade de dispositivos físicos, como eletrodomésticos, câmeras de vigilância, sensores de monitoramento, veículos, dentre outros, a IoT irá permitir o desenvolvimento de aplicações em diferentes domínios, dentre eles, a automação domiciliar, automação industrial, assistência para idosos, aplicações médicas, controle de trânsito e diversas outras. Estas aplicações farão uso da grande quantidade de dados gerados por estes dispositivos para criar novos tipos de serviços altamente dinâmicos e que requerem a mínima intervenção do usuário para o seu correto funcionamento (RAZZAQUE et al., 2016).

A ciência de situação é um aspecto central para que as aplicações possam lidar com as características heterogêneas de dados de diferentes fontes, precisões e disponibilidade espaço-temporal encontrados na IoT, permitindo a composição de uma visão em alto nível de abstração dos contextos de interesse das aplicações. O processamento dos dados de contexto pode envolver o emprego de diversas técnicas, baseadas em regras, aprendizado ou ambas, formando um fluxo de processamento em que informações são agregadas até que as situações de interesse possam ser identificadas (YE; DOBSON; MCKEEVER, 2012).

Nesse cenário, esta dissertação tem como objetivo geral a concepção de uma arquitetura de software aplicada nas etapas inerentes ao processamento contextual para a identificação de situações na IoT.

A arquitetura terá como premissa a integração com o *middleware* EXEHDA, contribuindo com o seu Subsistema de Adaptação e Reconhecimento do Contexto. Mais especificamente, os objetivos desta dissertação são:

- analisar trabalhos científicos relacionados com o tema de pesquisa proposto, sistematizando o estado da arte na área;
- conceber um modelo arquitetural que permita o gerenciamento e composição de fluxos de processamento contextuais híbridos ou não;
- explorar o processamento híbrido de contexto, pela combinação de componentes de software que contemplem diferentes técnicas de processamento contextual;
- adicionar suporte para a interação do EXEHDA com fontes de dados e serviços de processamento disponíveis na IoT;

- criar *Application Programming Interface*s (APIs) que permitam o acesso à arquitetura desenvolvida, possibilitando a parametrização necessária em diferentes domínios de aplicação;
- contribuir com dois estudos de caso na área de saúde, contemplando aspectos relevantes da área de IoT. Um aplicado em reabilitação cardíaca e outro em ambientes hospitalares;
- divulgar os resultados obtidos na pesquisa por meio da publicação de artigos científicos.

1.3 Apresentação do texto

Após este capítulo de introdução onde as principais motivações e objetivos do trabalho foram apresentados, o texto desta dissertação está estruturado em mais cinco capítulos, descritos a seguir.

No capítulo 2 são abordados os tópicos de interesse na área de Internet das Coisas e ciência de situação, também é apresentado o *middleware* EXEHDA e os seus principais componentes.

A revisão de literatura realizada acerca dos trabalhos relacionados, destacando tendências e lacunas de pesquisa é apresentada no capítulo 3.

No capítulo 4 é apresentada a arquitetura do EXEHDA-IS, caracterizando a abordagem para gerenciamento e composição de mecanismos para processamento híbrido de contexto aplicado na identificação de situações.

No capítulo 5 são apresentados os estudos de caso utilizados para avaliação da arquitetura proposta, buscando demonstrar como pode ser feito o desenvolvimento de aplicações cientes de situação na IoT utilizando o EXEHDA-IS.

Por fim, no capítulo 6 são apresentadas as considerações finais desta dissertação, destacando as principais conclusões, publicações realizadas e as propostas para continuidade da pesquisa.

2 ESCOPO DO TRABALHO

Neste capítulo é apresentada a revisão de literatura sobre os principais fundamentos conceituais envolvidos na concepção do EXEHDA-IS. Na primeira seção são apresentadas definições e temas relevantes em Internet das Coisas. Na segunda seção é introduzido o conceito de ciência de situação e também são apresentadas algumas técnicas empregadas no processamento contextual em aplicações cientes de situação. Na terceira e última seção é apresentado o *middleware* EXEHDA, destacando os principais componentes da sua arquitetura e o tipo de ambiente em que o mesmo é aplicado.

2.1 Internet das Coisas

Avanços em tecnologias de sensores, atuadores e protocolos de rede sem fio tem permitido o desenvolvimento de soluções de computação embarcada para diferentes finalidades. Dispositivos como utensílios domésticos, carros, equipamentos médicos e *smartphones* podem ser conectados e interagir a qualquer momento, em qualquer lugar com qualquer pessoa ou outro dispositivo através de uma rede unificada, denominada de Internet das Coisas – *Internet of Things* (IoT) (RAZZAQUE et al., 2016).

Não existe atualmente uma definição única para IoT, isso se deve ao fato desta ser uma área de pesquisa recente e também por sua aplicação ocorrer em diferentes áreas, nas quais são originadas definições próprias. Perera et al. (2013) destacam como sendo uma definição abrangente da área o conceito estabelecido por Vermesan et al. (2009), em que afirmam: "*A Internet das coisas permite que pessoas e coisas possam se conectar a qualquer hora, em qualquer lugar, com qualquer coisa, utilizando qualquer caminho, rede ou serviço*".

Na tentativa de estabelecer uma definição para a IoT, Minerva, Biru and Rotondi (2015) apresentam uma revisão sobre diversos termos e técnicas empregadas nessa área, estabelecendo ao final do trabalho as principais funções e cenários de uso esperados para a IoT. Os autores apresentam duas definições para IoT, baseadas essencialmente na escala do sistema. Estas definições serão transcritas a seguir:

Cenários de aplicação de pequena escala

"*IoT é a rede que conecta "Coisas" unicamente identificáveis na internet. Estas "Coisas" possuem capacidade de sensoriar/atuar e de serem programadas. Através do identificador único, informações podem ser coletadas e comandos de atuação podem ser gerados de qualquer lugar, a qualquer momento por qualquer coisa*".

Cenários de aplicação de larga escala

"*A IoT prevê uma autoconfigurável, adaptativa e complexa rede que interconecta "Coisas" através da internet utilizando protocolos de comunicação padronizados. As "Coisas" possuem representação física ou virtual no mundo digital, capacidade de sensoriamento/atuação, são programáveis e possuem identificação única. A sua representação contém informações que incluem a identidade, estado, localização ou outras informações relevantes para as aplicações. As "Coisas" oferecem serviços com ou sem a intervenção humana. Os serviços são explorados através de interfaces inteligentes que estão disponíveis em qualquer lugar, a qualquer momento, para qualquer coisa considerando aspectos de segurança*".

Na Figura 2.1 está representada uma abstração dos cenários de uso da IoT, onde elementos de naturezas diversas interagem através de uma rede unificada empregando a internet, possibilitando a construção de aplicações inovadoras que exploram o dinamismo desse ambiente conectado. Minerva, Biru and Rotondi (2015) definem os mecanismos operacionais listados a seguir como fundamentais para as aplicações em IoT:

- **Interconexão**: a primeira característica derivada da IoT é a possibilidade de interconexão dos dispositivos do meio físico relevantes para o usuário e a aplicação;
- **Conexão com a internet**: a internet é utilizada como meio para conexão e disponibilização dos elementos;
- **Identificação única**: um sistema de IoT é composto por dispositivos com identificação única;
- **Ubiquidade**: as aplicações de IoT devem estar disponíveis de forma oportuna em qualquer lugar e a qualquer momento;
- **Capacidade de sensoriamento e atuação**: sensores e atuadores ligados as coisas permitem que estas tenham um comportamento mais inteligente;

- **Inteligência integrada**: objetos dinâmicos com inteligência integrada tornam-se extensões do corpo e da mente humana;
- **Interoperabilidade**: a comunicação na IoT deve ser feita a partir de padrões que permitam a comunicação entre os diversos tipos de dispositivos conectados;
- **Autoconfiguração**: devido ao grande número de elementos conectados, a gerência individual de cada dispositivo torna-se impraticável, desta forma a tendência na IoT é que as configurações de hardware e software necessárias para a correta operação sejam feitas de forma autônoma pelos dispositivos;
- **Programabilidade**: os dispositivos devem oferecer a capacidade de programação, permitindo a adaptação do seu funcionamento sem a necessidade de alterações físicas.

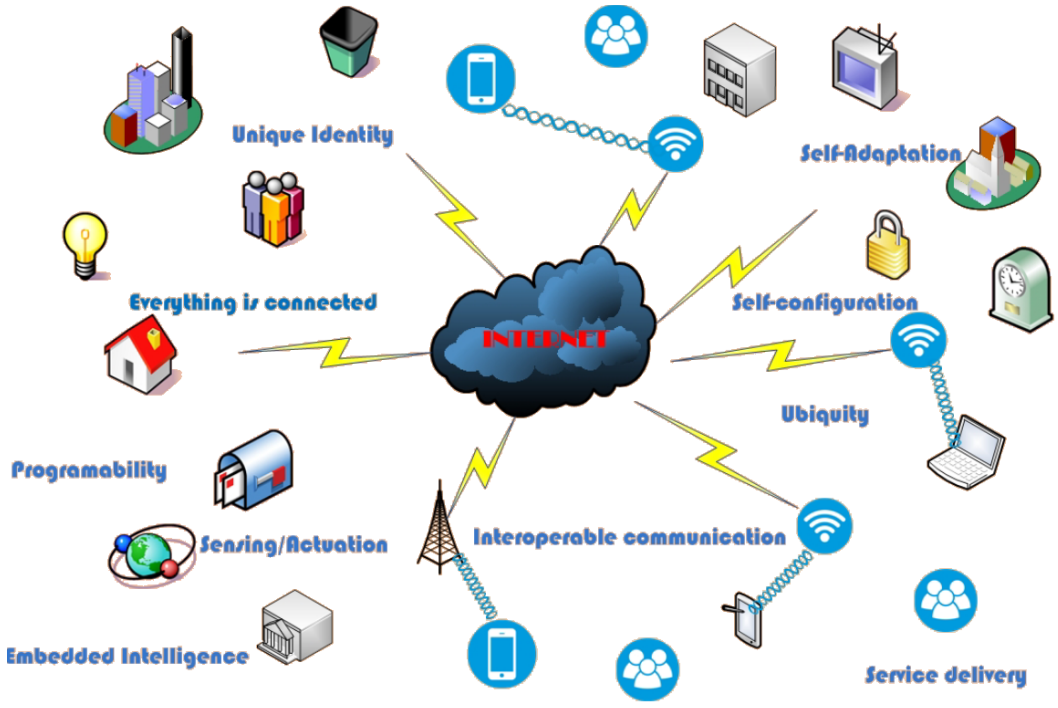

Figura 2.1 – Abstração das funcionalidades e cenários de uso da IoT.
Fonte: (MINERVA; BIRU; ROTONDI, 2015).

O crescimento do número de dispositivos conectados, propiciado pelos avanços em IoT, fez com que a quantidade de dados coletados e disponibilizados para as aplicações crescesse significativamente. Porém, para muitas aplicações os contextos de interesse são obtidos através da fusão de dados de sensores, estabelecendo uma visão mais abstrata do contexto atual, permitindo a inferência da situação do usuário e do próprio sistema. Nesse cenário, um dos

principais desafios de pesquisa em IoT é a ciência de situação (PERERA et al., 2013). Este tema é abordado na próxima seção.

2.2 Ciência de Situação

A ciência de situação representa o entendimento do ambiente, incluindo parâmetros da própria aplicação, representados através de informações contextuais das entidades em intervalos de espaço e tempo, propiciando a interpretação de significado e projeção em ações futuras (ENDSLEY, 1995).

Em computação, uma situação corresponde a uma visão compreensível e em alto nível de abstração do contexto de interesse das aplicações. Esta visão é resultante de dados de contexto coletados de sensores distribuídos pelo ambiente (BELLAVISTA et al., 2012; KNAPPMEYER et al., 2013). As situações das entidades podem ser obtidas através do processamento contextual, agregando significado as informações coletadas de sensores ou de outras fontes de dados relevantes (COSTA et al., 2006).

Na Figura 2.2 pode ser visto um exemplo de processamento contextual aplicado à identificação de situações, no qual é mostrada a inferência de uma situação de risco de saúde para um dado usuário de um sistema. Esta inferência é feita através do processamento contextual dos dados coletados de um sensor de frequência cardíaca, de um acelerômetro e de um GPS. Os dados dos sensores são convertidos para domínios linguísticos interpretáveis e que permitem a construção de modelos de raciocínio baseados no conhecimento da área de aplicação.

Segundo Ye, Dobson and McKeever (2012), a definição de uma situação e o seu refinamento pode ser feito através de relações com outras situações ou variáveis de contexto. Em Ye, Dobson and McKeever (2012) são definidos ainda os principais tipos de relações que podem ser estabelecidas entre as situações, conforme descrito a seguir:

- **generalização:** uma situação pode ser considera mais geral do que outra. Por exemplo, a situação “exercitando-se” generaliza a situação “jogando futebol”;
- **composição:** uma situação pode ser composta por situações menores. Por exemplo, “jogando futebol” pode ser composto por “correndo” e “chutando”;
- **dependência:** quando a ocorrência de uma situação depende da outra. Por exemplo, “comemorando o próprio gol” depende da situação “jogando futebol”;

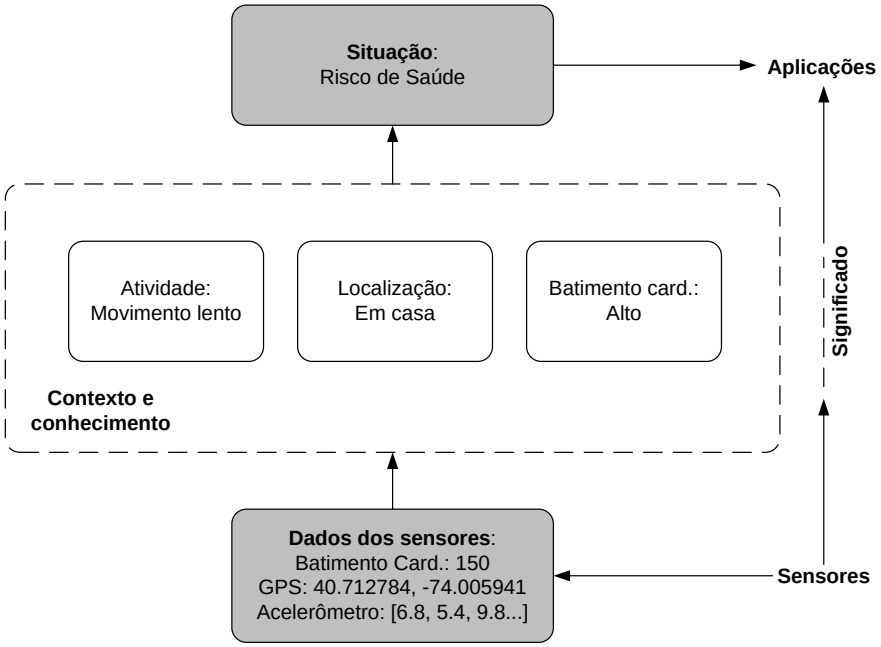

Figura 2.2 – Exemplo de processamento contextual aplicado na identificação de situações.
Fonte: adaptado de (YE; DOBSON; MCKEEVER, 2012).

- **contradição:** duas situações que não podem ocorrer para uma mesma entidade, no mesmo local e ao mesmo tempo. Por exemplo, as situações "jogando futebol" e "jogando tênis" são mutuamente exclusivas;
- **sequência temporal:** uma dada situação pode ocorrer antes, depois ou ainda de forma intercalada com outra. Por exemplo, a situação "discutindo com o árbitro" pode ser antecedida por "jogando futebol" e sucedida por "indo para o vestiário".

Considerando a necessidade de tratar estas diferentes relações, diversas técnicas têm sido consideradas, e seu estudo foi central para o desenvolvimento do EXEHDA-IS. Pode-se dizer que as técnicas utilizadas para prover ciência de situação aos sistemas estão divididas em dois grupos. Aquelas baseadas em especificação e as baseadas em aprendizado (YE; DOBSON; MCKEEVER, 2012). Estes grupos podem ainda ser subdivididos em alguns subconjuntos de técnicas. Algumas destas, consideradas promissoras por Ye, Dobson and McKeever (2012), serão apresentadas a seguir.

2.2.1 Técnicas Baseadas em Especificação

As técnicas baseadas em especificação são geralmente aplicadas em sistemas que possuem um número reduzido de sensores, para os quais os dados e relações podem ser interpretados através de regras elaboradas por especialistas (YE; DOBSON; MCKEEVER, 2012).

Lógica Formal

A utilização de lógica formal para modelagem de situação depende da capacidade de modularização das situações e discretização de variáveis de contexto (YE; DOBSON; MCKEEVER, 2012). A lógica formal é constituída por um conjunto de operações matemáticas definidas principalmente em teoria de conjuntos e álgebra booleana. Como ferramenta de programação para lógica formal, pode ser utilizada a linguagem Prolog (LOKE, 2004).

No código 2.3 pode ser visto um exemplo de modelagem de ciência de situação, aplicando lógica formal modelada em Prolog. No mesmo, é feita a configuração do tipo de toque de um telefone celular baseada na situação do usuário. Para as situações "reunião" (*meeting*) e "palestra" (*lecture*), o toque é configurado para silencioso. Quando o usuário estiver em um restaurante (*restaurant*), os sons de alerta do celular são habilitados.

```
required_phone_mode(quiet) :- my_current_user(E), in_meeting_now*>E.
required_phone_mode(quiet) :- my_current_user(E), in_lecture*>E.
required_phone_mode(noisy) :- my_current_user(E), restaurant*>E.
```

Figura 2.3 – Exemplo de modelagem de ciência de situação utilizando a linguagem Prolog.
Fonte: (LOKE, 2004).

Ontologia

Modelos baseados em ontologias tiram proveito da capacidade destas em expressar relações complexas. A validade de dados é geralmente expressa através da imposição de restrições, centrando-se sobre as relações entre as entidades. Ontologias são adequadas para mapear o conhecimento dentro de uma estrutura de dados facilmente utilizada e interpretada. Além disso, a ampla adoção da ontologia permite a reutilização de trabalhos anteriores e a criação de vocabulários de domínio comuns e compartilhados (BELLAVISTA et al., 2012; YE; DOBSON; MCKEEVER, 2012).

Na Figura 2.4 pode ser visto um exemplo de inferência para situação de uma entidade baseado na modelagem por ontologia. Trata-se da situação da instância da classe "Humano", definida como "João" que possui a variável de contexto "temperatura", com valor igual a 39,0 ^{o}C. Como o valor da temperatura é maior do que o definido na relação de dependência, a propriedade inferida "Febre" recebe o estado "Sim". Logo, pode-se afirmar: "João está com febre".

Devido à simplicidade e limitado número de variáveis, a inferência de situação para o exemplo da Figura 2.4 pode ser facilmente modelada através de programação sequencial utilizando lógica booleana em conjunto com um modelo de ontologia. No entanto, em muitas aplicações, a quantidade de variáveis de contexto e entidades que precisam ser correlacionadas para inferir situações pode ser maior. Aspectos como incerteza de medições, disponibilidade incerta de recursos, dependência espacial e temporal e requisito de autoaprendizado, fazem com que técnicas mais abrangentes sejam empregadas na inferência de situação (YE; DOBSON; MCKEEVER, 2012).

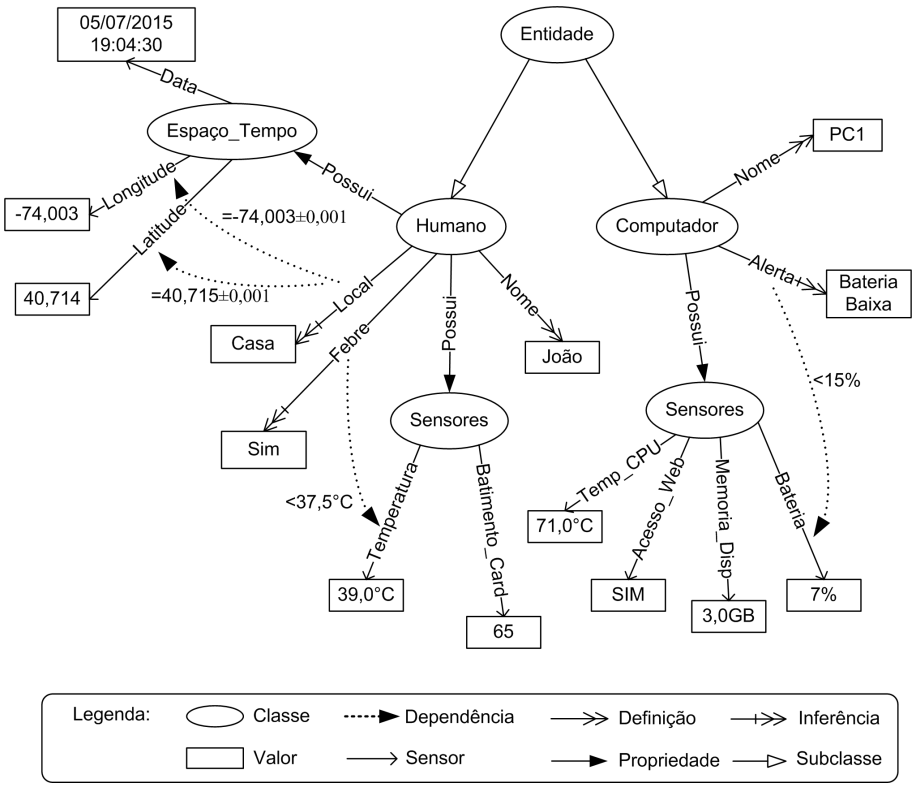

Figura 2.4 – Exemplo de modelagem de contexto baseada em ontologia.

Dentre as ferramentas disponíveis para modelagem gráfica de ontologias, pode ser citado o software *Protégé*, desenvolvido na Universidade de Stanford. O mesmo permite a geração de código para *Web Ontology Language* (OWL), a partir de um modelo criado graficamente. Na Figura 2.5 tem-se um exemplo de ontologia gerado nesta ferramenta. A descrição em OWL exportada para a mesma pode ser vista no código mostrado na Figura 2.6.

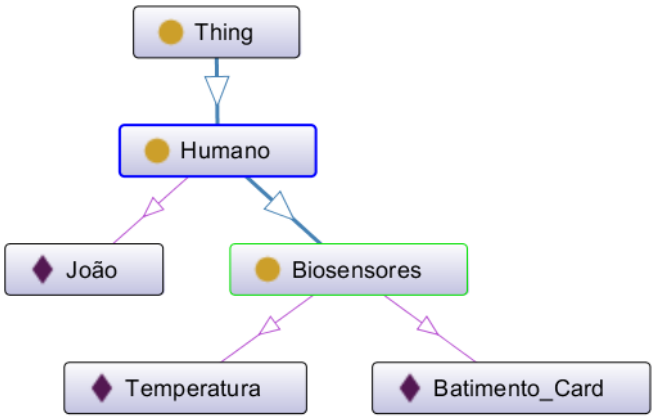

Figura 2.5 – Exemplo de modelo de ontologia criado no software *Protégé*.

Lógica Espaço-temporal

Lógica espaço-temporal é uma área bem estabelecida em Inteligência Artificial (IA), aplicada na representação e raciocínio sobre as características e restrições de contexto e situações (ALLEN; FERGUSON, 1994). Allen and Ferguson (1994) propuseram um método para representação e raciocínio lógico para modelagem de problemas que dependem de informações de espaço e tempo. Allen sugeriu que a representação de tempo e espaço por intervalos poderia ser mais interessante do que a representação por pontos específicos.

No trabalho (AUGUSTO et al., 2008) foram aplicados conceitos de modelagem espaço-temporal para ambientes inteligentes. No código 2.7 pode ser visto um exemplo de modelagem extraído deste trabalho. O operador de alto nível *ANDlater*, representa a sucessão de um evento temporal.

Lógica Fuzzy

Em aplicações baseadas em IoT, a quantidade e a heterogeneidade de sensores manipulados têm crescido. A incerteza de disponibilidade e diferentes faixas de precisão de sensores agregam incertezas ao sistema. Além disso, na modelagem de contexto e de situação, é necessário gerar inferências de estados a partir de dados de sensores. Em muitos casos, essa classificação não está relacionada com um valor especifico, como falso ou verdadeiro, mas sim com uma faixa de valores (YE; DOBSON; MCKEEVER, 2012; PERERA et al., 2013). Nesse tipo de cenário, a lógica fuzzy ou difusa tem se mostrado uma técnica promissora para tratar a incerteza de dados e mapear saídas consistentes para ao sistema (SHARMA, 2012).

```
<Ontology xmlns="http://www.w3.org/2002/07/owl#"
    <Declaration>
    <Class IRI="#Biosensores"/>
    </Declaration>
    <Declaration>
        <Class IRI="#Humano"/>
    </Declaration>
    <Declaration>
        <NamedIndividual IRI="#Batimento_Card"/>
    </Declaration>
    <Declaration>
        <NamedIndividual IRI="#João"/>
    </Declaration>
    <Declaration>
        <NamedIndividual IRI="#Temperatura"/>
    </Declaration>
    <SubClassOf>
        <Class IRI="#Biosensores"/>
        <Class IRI="#Humano"/>
    </SubClassOf>
    <ClassAssertion>
        <Class IRI="#Biosensores"/>
        <NamedIndividual IRI="#Batimento_Card"/>
    </ClassAssertion>
    <ClassAssertion>
        <Class IRI="#Humano"/>
        <NamedIndividual IRI="#João"/>
    </ClassAssertion>
    <ClassAssertion>
        <Class IRI="#Biosensores"/>
        <NamedIndividual IRI="#Temperatura"/>
    </ClassAssertion>
</Ontology>
<!-- Generated by the OWL API (version 3.5.1) http://owlapi.sourceforge.net -->
```

Figura 2.6 – Código OWL gerado através do software *Protégé* para o modelo de ontologia mostrado na Figura 2.5.

A lógica fuzzy, ao contrário da lógica booleana, permite respostas intermediárias entre falso e verdadeiro, ou seja, assume a existência de infinitos valores ente 0 e 1. Assim, conceitos intrínsecos da linguagem humana para representar situações de forma aproximada podem ser modelados. Por exemplo, as afirmações: “está pouco frio” ou “está muito quente”, de acordo com o contexto, podem transmitir de maneira satisfatória a informação entre os interlocutores, sem a necessidade de tratar valores numéricos exatos ou escalas de medidas.

Na computação ubíqua, a lógica fuzzy pode ser utilizada como ferramenta na inferência de situação através do mapeamento do grau de pertinência de uma variável para a ocorrência de uma situação (HAGHIGHI et al., 2014). Isso é feito através de variáveis linguísticas, as quais podem assumir valores de um determinado conjunto de termos. A definição dos conjuntos para as variáveis linguísticas pode ser feita com o auxílio de uma ferramenta para ontologia.

Na Figura 2.8 pode ser visto um exemplo de funções de pertinência para realizar o

```
IF at_kitchen_on ANDlater
       tdRK_on ANDlater
       no_movement_detected
THEN assume the occupant has fainted
```

Figura 2.7 – Lógica espaço-temporal aplicada no monitoramento de um indivíduo em um ambiente inteligente.

Fonte: (AUGUSTO et al., 2008).

mapeamento da variável temperatura para o domínio linguístico que representa os níveis de febre, a qual pode assumir os valores: “Baixa”, “Média” e “Alta”. Como trata-se de uma técnica baseada em especificação, as faixas de valores são estabelecidas a partir do conhecimento de especialistas. Porém, inferências como “febre um pouco alta” para 38,5 ºC e “febre muita alta” para 40 ºC, podem ser facilmente geradas.

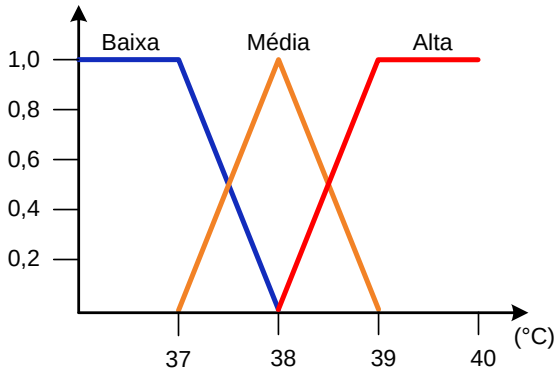

Figura 2.8 – Funções de pertinência para modelagem do conceito de febre utilizando lógica *fuzzy*.

Quando uma dada situação depender de mais de uma variável, a pertinência final pode ser dada pelo somatório da pertinência de cada variável multiplicada por um peso correspondente à sua contribuição global na situação, conforme expresso na equação 2.1 (HAGHIGHI et al., 2014).

$$P = \sum_{i=1}^{N} w_i.\mu(x_i) \tag{2.1}$$

Onde: x_i representa uma dada variável contínua ou discreta, $\mu(x_i)$ representa a função de pertinência individual da variável, e w_i representa o peso da variável na situação final.

2.2.2 Técnicas Baseadas em Aprendizado

As técnicas baseadas em aprendizado permitem que o sistema gere inferências de forma automática, estabelecendo relações entre dados de sensores e estados anteriores do sistema através de um processo de aprendizado de máquina. Essas técnicas são aplicadas em sistemas em que existem muitos dados de entradas e sensores de forma que a inferência de situação baseada em especificação torna-se inviável (YE; DOBSON; MCKEEVER, 2012). Algumas das técnicas baseadas em aprendizado, aplicadas na área de processamento contextual e identificação de situações são descritas a seguir.

Redes Bayesianas

Uma Rede Bayesiana pode ser representada por um gráfico acíclico, em que cada nó representa uma variável que pode ser discreta ou contínua, e cada arco é a relação causal entre os nós. Se houver um arco a partir de um nó A para outro nó B, então A é chamado um pai de B, o que implica que a variável B é considerado como diretamente dependente de A (YE; DOBSON; MCKEEVER, 2012).

Na Figura 2.9 tem-se uma exemplo de rede de classificação Bayesiana, a partir da qual pode ser calculada a probabilidade da variável "C" em assumir um dado estado "j" em função das probabilidades condicionais P($X_i|C$), conforme equação 2.2.

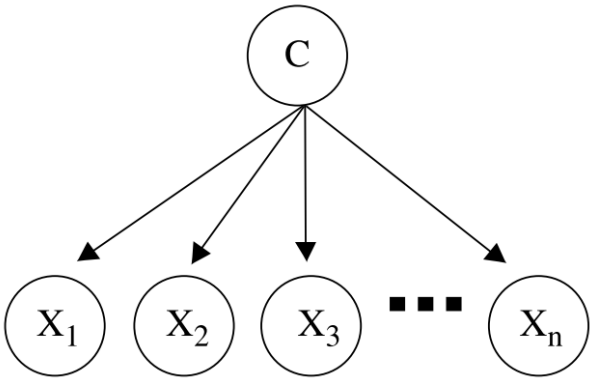

Figura 2.9 – Estrutura de uma rede de classificação Bayesiana.
Fonte: (KORPIPÄÄ et al., 2003).

$$P(C_j|X_1...X_n) = \prod_{i=1}^{n} P(X_i|C) \tag{2.2}$$

O processo de aprendizado de uma rede Bayesiana consiste em determinar o mapa de probabilidades condicionais $P(X_i|C)$ para cada variável da rede, possibilitando assim que, a partir dos valores de um dado conjunto de variáveis, a probabilidade condicional de outra possa ser determinada seguindo a equação 2.2 (KORPIPÄÄ et al., 2003).

Árvores de Decisão

Árvore de decisão é um método de classificação preditivo no qual condições de teste e resultados são organizados em uma estrutura de árvore. Para cada execução do algoritmo é escolhido um valor para uma dada variável. Durante a execução, várias classes de variáveis podem ser utilizadas, no entanto, a folha final de cada ramo da árvore deve pertencer a uma mesma classe, a qual representa o valor de classificação desejado. Uma vantagem desse método é a sua facilidade de implementação e entendimento (YE; DOBSON; MCKEEVER, 2012; WITTEN; FRANK; HALL, 2011). Existem na literatura vários métodos que podem ser empregados na modelagem e treinamento de árvores de decisão, um método amplamente utilizado é o C4.5 (WITTEN; FRANK; HALL, 2011).

O software Weka [1] disponibiliza uma série de algoritmos para classificação e mineração de dados, dentre eles o C4.5. Aplicando este método sob um arquivo de treinamento no formato *Attribute-Relation File Format* (ARFF) mostrado no código 2.10, foi obtida a árvore de decisão mostrada na Figura 2.11. No exemplo em questão, está modelada a probabilidade para ocorrência de um jogo de tênis baseado em um histórico de eventos relacionados com condições climáticas.

[1] Para mais informações acesse: http://www.cs.waikato.ac.nz/ml/weka/

```
@relation jogo.simbolico

@attribute tempo {ensolarado, nublado, chuvoso}
@attribute temperatura {quente, média, fria}
@attribute umidade {alta, normal}
@attribute vento {sim, não}
@attribute jogar {sim, não}

@data
ensolarado,quente,alta,não,não
ensolarado,quente,alta,sim,não
nublado,quente,alta,não,sim
chuvoso,média,alta,não,sim
chuvoso,fria,normal,não,sim
chuvoso,fria,normal,sim,não
nublado,fria,normal,sim,sim
ensolarado,média,alta,não,não
ensolarado,fria,normal,não,sim
chuvoso,média,normal,não,sim
ensolarado,média,normal,sim,sim
nublado,média,alta,sim,sim
nublado,quente,normal,não,sim
chuvoso,média,alta,sim,não
```

Figura 2.10 – Arquivo ARFF utilizado para modelagem e treinamento da árvore de decisão.
Fonte: adaptado da base de exemplos do software Weka.

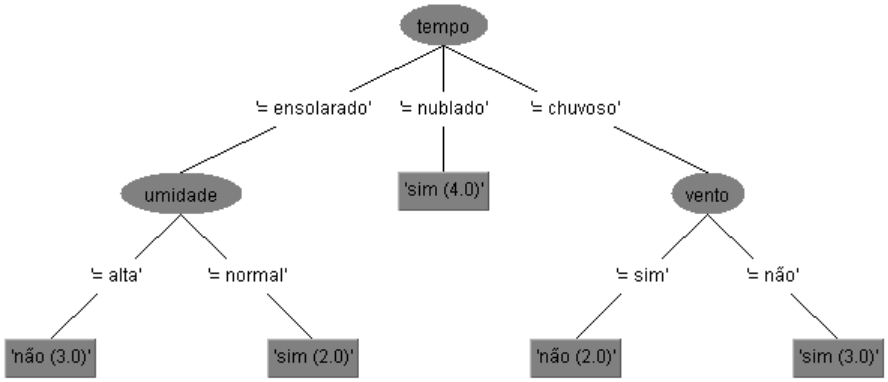

Figura 2.11 – Árvore de decisão para previsão da ocorrência de um jogo de tênis baseada em condições climáticas.

Modelos Ocultos Markov

Uma cadeia de Markov é um caso particular de processo estocástico em que estados discretos podem ser identificados e que a probabilidade de transição do estado atual para um estado futuro depende somente do estado atual (FOSLER-LUSSIER, 1998; RABINER, 1989). Na Figura 2.12 pode ser visto um exemplo de cadeia de Markov para previsão do tempo baseada em probabilidade.

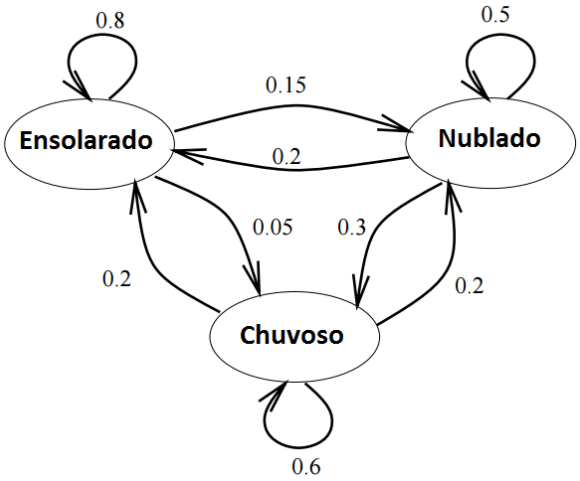

Figura 2.12 – Exemplo de cadeia de Markov para modelagem de previsão do tempo baseada em probabilidade.

Fonte: adaptado de: (FOSLER-LUSSIER, 1998).

No Modelo Oculto de Markov é assumido que o processo em análise apresenta as propriedades estocásticas do Modelo de Markov, porém este não é diretamente observável. O mesmo pode ser inferido através de um outro conjunto de variáveis estocásticas que produzem uma sequência de observações independentes entre si (RABINER, 1989).

Redes Neurais

Redes neurais artificiais são uma técnica sub-simbólica, originalmente inspirada por redes de neurônios biológicos. As redes neurais podem aprender automaticamente e mapear complexas relações não-lineares. Uma rede neural é composta de muitos neurônios artificiais que estão ligados entre si de acordo com uma arquitetura de rede específica (YANG; WANG; CHEN, 2008). Na Figura 2.13 pode ser visto um exemplo de arquitetura para um classificador baseado em uma rede neural artificial.

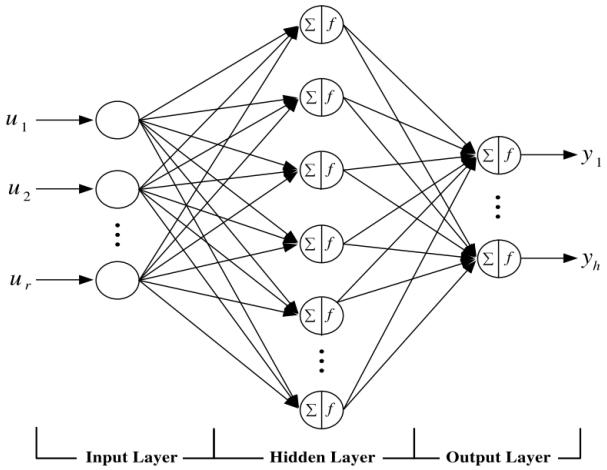

Figura 2.13 – Exemplo de classificador baseado em uma rede neural.
Fonte: (YANG; WANG; CHEN, 2008).

O desempenho de redes neurais é afetado pela quantidade de dados de treinamento. Uma rede neural é considerada uma boa opção se há abundância de dados para treinamento e se o domínio do problema é mal compreendido para derivar um modelo aproximado (YE; DOBSON; MCKEEVER, 2012).

2.3 Middleware EXEHDA

A arquitetura de software explorada neste dissertação tem como premissa a sua integração com o *middleware* EXEHDA (YAMIN, 2004; LOPES et al., 2014a). Nesta seção é apresentada uma revisão da arquitetura e das principais funções desse *middleware*.

O EXEHDA é um *middleware* para computação ubíqua e suas implementações como a IoT. O mesmo é baseado em serviços, sendo aplicado no gerenciamento do ambiente ubíquo e utilizado para criação de aplicações sobre esse ambiente. A sua arquitetura é concebida de forma distribuída e propõe o uso da semântica siga-me, possibilitando que as aplicações estejam disponíveis para o usuário a partir de qualquer lugar, todo o tempo.

Existem dois tipos de servidores na arquitetura do EXEHDA: (i) Servidor de Borda, responsável por interagir com ambiente através de sensores e atuadores; e (ii) Servidor de Contexto, responsável por prover funcionalidades para ciência de situação. Esses servidores são alocados em células do ambiente gerenciado pelo EXEHDA, onde cada célula possui um Servidor de Contexto e pode possuir vários Servidores de Borda. Na Figura 2.14 pode ser vista a disposição destes servidores no ambiente.

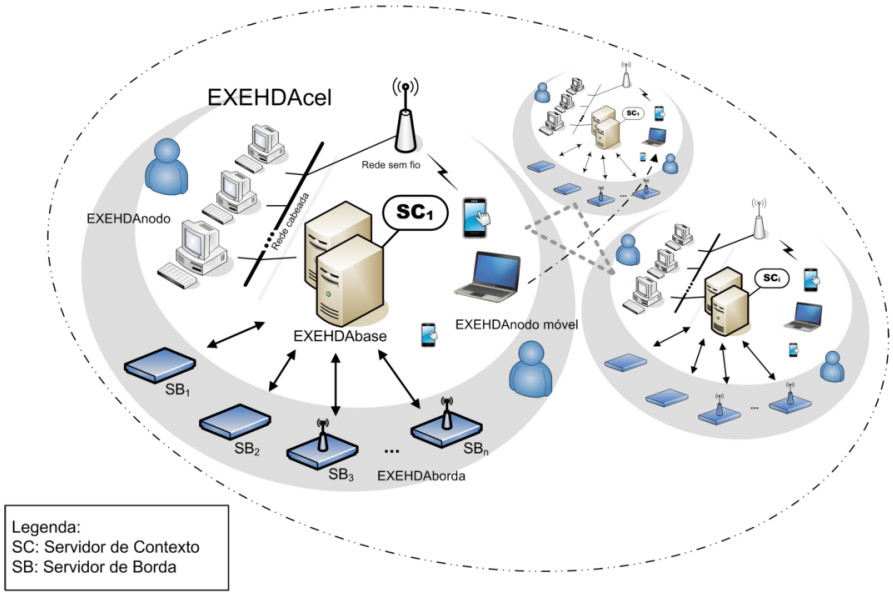

Figura 2.14 – Ambiente gerenciado pelo EXEHDA.
Fonte: (LOPES et al., 2014b).

A arquitetura de software do EXEHDA é organizada em três camadas, conforme Figura 2.15. A camada inferior mostrada na figura engloba componentes de hardware e software de suporte para a execução do *middleware* e acesso ao meio físico. A camada superior é formada por um *framework* para programação das aplicações. A parte central da arquitetura representa os componentes do EXEHDA. Dentre os componentes do *middleware*, destaca-se a existência de quatro subsistemas, descritos a seguir:

- **Subsistema de Execução Distribuída**: agrupa uma série de serviços que permitem a execução distribuída das aplicações sobre o ambiente ubíquo. Possui uma abstração de objetos chamada OX (Objeto eXehda). Esses objetos utilizam um conjunto de meta-informações que permitem a sua criação e acesso em qualquer nodo do *middleware*;

- **Subsistema de Comunicação**: permite a comunicação entre componentes distribuídos da aplicação, oferece um mecanismo para chamada remota de procedimentos semelhante ao *Remote Method Invocation* (RMI), porém sem exigir a manutenção da conexão durante toda a execução da chamada. Além disso, dispõem de mecanismos para desacoplamento temporal e espacial baseado em espaços de tuplas;

- **Subsistema de Acesso Ubíquo**: componente em que são gerenciadas as bases de dados das aplicações, controle de acesso de usuários, controle da mobilidade de dados e código para implementação de funções para compor a semântica siga-me;

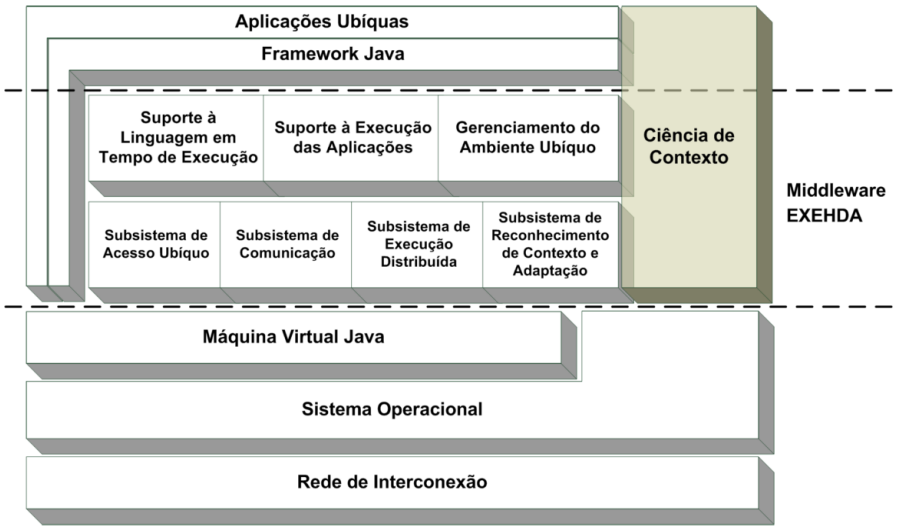

Figura 2.15 – Arquitetura de software do EXEHDA.
Fonte: (LOPES et al., 2014a).

- **Subsistema de Reconhecimento de Contexto e Adaptação**: este subsistema possui funções para processamento contextual, permitindo que, a partir de dados brutos obtidos dos servidores de borda, possam ser geradas informações contextuais em alto nível de abstração, possibilitando a construção de aplicações cientes de situação. Além disso, esse subsistema gera as ações de adaptação sobre o ambiente ubíquo baseado nas informações contextuais obtidas.

2.4 Considerações Sobre o Capítulo

Neste capítulo foram apresentadas as definições mais recentes disponíveis na literatura para a área de IoT, nota-se que esta é uma área que abrange diversas frentes de pesquisa, buscando criar um ambiente onde pessoas e coisas possam interagir em qualquer local e a qualquer momento, propiciando o surgimento de aplicações antes inconcebíveis.

Foi apresentado o conceito de ciência de situação e destacou-se técnicas para processamento contextual difundidas na área de IoT. Estas técnicas estão divididas em duas classes principais: especificação e aprendizado.

Por fim foram apresentadas as características e módulos do *middleware* EXEHDA, utilizado nesta dissertação como solução arquitetural para o desenvolvimento de aplicações em IoT.

3 TRABALHOS RELACIONADOS

Neste capítulo é apresentada a revisão de literatura realizada para identificar os trabalhos relacionados ao tema pesquisado nesta dissertação. As principais características dos trabalhos selecionados estão resumidas neste capítulo. Ao final do mesmo é apresentada uma discussão sobre os trabalhos, contrapondo as suas características e evidenciando tendências e lacunas de pesquisa, utilizadas como subsídio para composição da proposta desta dissertação.

3.1 Projeto CARA

O projeto CARA - *Context-Aware Real-time Assistant* (YUAN; HERBERT, 2014) tem por objetivo a criação de um sistema para prover serviços personalizados de assistência remota para idosos, permitindo a adaptação do sistema de acordo com as atividades normais do usuário. O mesmo prevê o uso de um modelo híbrido para processamento contextual, no qual técnicas para especificação de regras são combinadas com um algoritmo de aprendizado baseado em casos.

O sistema é constituído por quatro componentes principais, descritos a seguir:

- **Sensores vestíveis**: dispositivos eletrônicos utilizados para monitorar os sinais vitais do paciente, que inclui sensores como ECG, temperatura e movimentação;
- **Sensores do ambiente**: sensores conectados em uma rede wireless no ambiente monitorado, incluindo sensores de presença, temperatura, dentre outros;
- **Sistema de processamento e monitoramento remoto**: sistema remoto responsável por receber os dados continuamente coletados pelos sensores vestíveis e do ambiente e executar o processamento contextual;
- **Sistema de dados e vídeo**: parte dos sistema desenvolvido para permitir um ambiente de consulta remota por vídeo e áudio.

Os quaro tipos de componentes são organizado arquiteturalmente e interagem conforme mostrado na Figura 3.1

O processamento contextual proposto no trabalho é feito através de um modelo híbrido que combina aprendizado baseado em casos com regras construídas em lógica fuzzy. Na Figura 3.2 podem ser vistas as três etapas principais no fluxo processamento contextual. São elas:

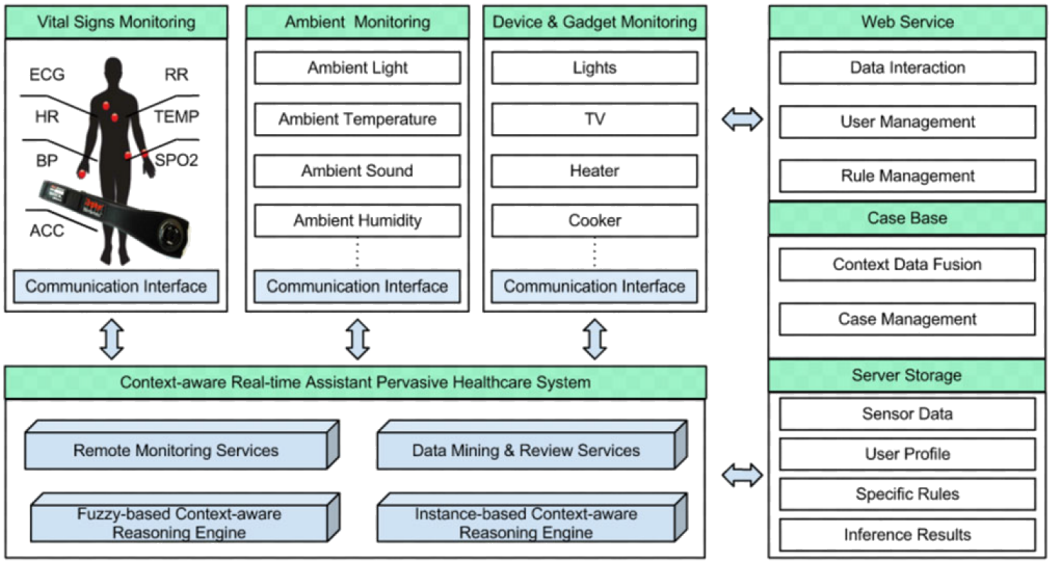

Figura 3.1 – Arquitetura proposta no projeto CARA.
Fonte: (YUAN; HERBERT, 2014).

(I) fusão de dados contextuais; (II) raciocínio baseado em casos e (III) raciocínio baseado em regras fuzzy. Nota-se que o módulo de processamento/raciocínio baseado em casos recebe duas entradas de realimentação: uma para atribuição dinâmica de pesos para novos casos detectados e outra que permite uma solicitação de revisão de casos. Os casos aprendidos são armazenados em um repositório acessível para o sistema de regras.

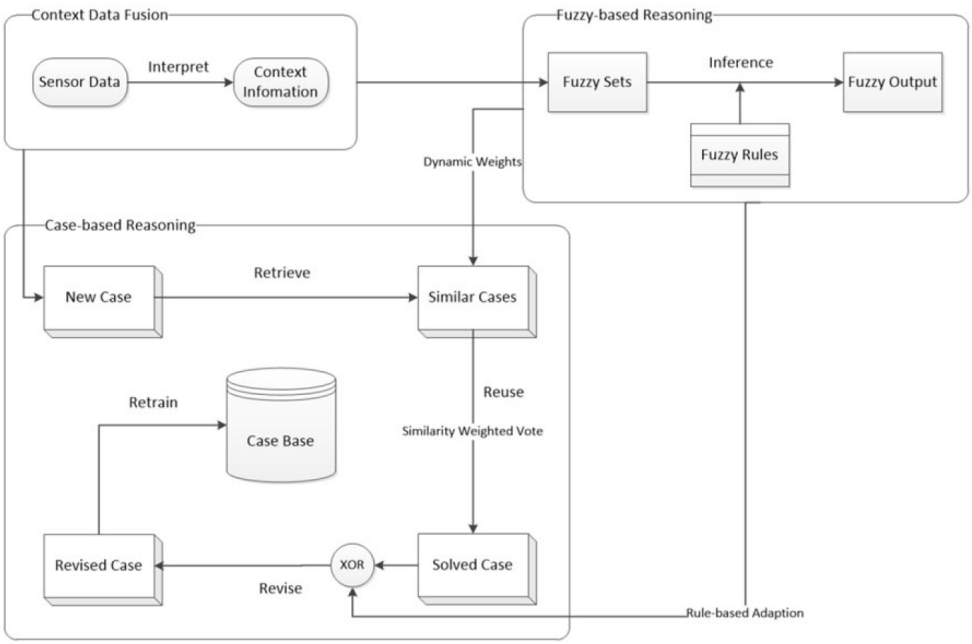

Figura 3.2 – Modelo de processamento híbrido proposto no projeto CARA.
Fonte: (YUAN; HERBERT, 2014).

3.2 Projeto Programmable Context Awareness Framework

No *framework* proposto por Chihani, Bertin and Crespi (2014) é concebido um modelo para programação do processamento contextual focado na concepção de aplicações cientes do contexto. Os autores destacam como diferencial do seu *framework* o desacoplamento do modelo de representação do contexto com a etapa de processamento. Os autores citam outros *frameworks* e arquiteturas em que é suposto um compartilhamento do modelo contextual, o que em muitos casos limita a sua utilização em diferentes domínios de aplicação.

Para executar o processamento contextual, é proposto no *framework* o uso de uma linguagem baseada na linguagem de marcação *Extensible Markup Language* (XML), denominada pelos autores como *Context Processing Definition Language* (CPDL). Através desta linguagem é possível criar fluxos de processamento contextual considerando os componentes mostrados na Figura 3.3, divididos em quatro categorias:

- ***Filter***: possui funções de processamento de sinais, como filtros de Kalman, passa-baixas e passa-altas;
- ***Abstract***: componentes responsáveis por elevar o nível de abstração dos sinais coletados. Esta abstração é baseada em uma máquina de estados finitos que representa todos os possíveis valores que uma variável de contexto pode assumir. Em cada alteração de valor desta variável é executada uma regra para avaliar se o estado deve ser alterado ou mantido;
- ***Select***: esta função permite que a fonte de contexto mais adequada seja selecionada, baseado na qualidade do contexto. Um exemplo desse tipo de seleção seria a escolha da fonte de localização para uma pessoa considerando o uso de um smartphone, que pode ser obtida através do GPS ou de informações sobre a conexão de internet. Nesse caso a escolha dependerá da precisão necessária para a aplicação;
- ***Aggregate***: último estágio do processamento contextual proposto no trabalho, em que regras do tipo IF-THEN podem ser criadas. Estas regras permitem que eventos de notificação possam ser gerados para as aplicações clientes do *framework*.

O *framework* proposto prevê o modelo de representação e acesso ao contexto mostrado na Figura 3.4. São previstos dois tipos de usuários: desenvolvedores e clientes das aplicações.

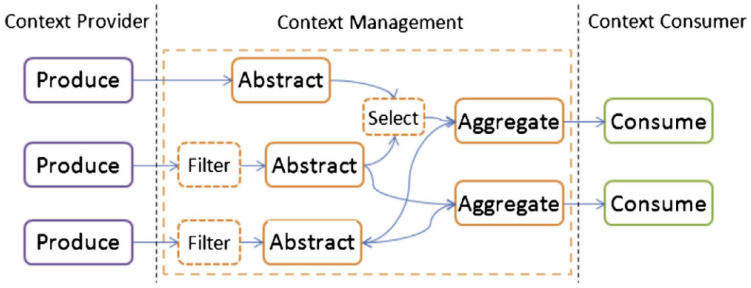

Figura 3.3 – *Framework* para processamento contextual programável proposto por Chihani, Bertin and Crespi (2014).

Fonte: (CHIHANI; BERTIN; CRESPI, 2014).

Os itens CxP, CxB e CxC representam provedores de contexto, *broker* de contexto e consumidores de contexto, respectivamente. Os *brokers* de contexto são responsáveis por prover o desacoplamento das variáveis de contexto do *framework* para os clientes da aplicação. Neste módulo são distribuídos para os consumidores de contexto os dados gerados pelos provedores.

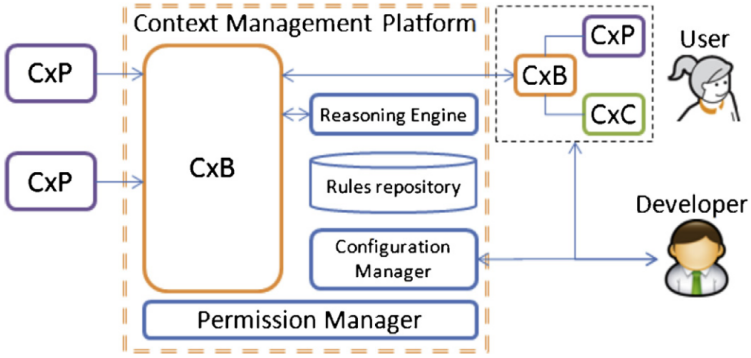

Figura 3.4 – Modelo de acesso e distribuição de contexto proposto no *framework* criado por Chihani, Bertin and Crespi (2014).

Fonte: (CHIHANI; BERTIN; CRESPI, 2014).

3.3 Projeto Simurgh

A arquitetura chamada Simurgh (KHODADADI; DASTJERDI; BUYYA, 2015) foi concebida com finalidade de permitir a descoberta de recursos, programação de fluxos de processamento e integração de serviços disponíveis na IoT. Os autores destacam como um problema central na área de IoT a dificuldade de integrar serviços disponíveis em diferentes plataformas comerciais ou abertas. Segundo os autores, isso se deve ao fato de ainda não existir na IoT um padrão compreensível e acessível através de um *framework* simples que permita o uso de dispositivos da IoT.

A arquitetura proposta é divida em duas camadas, *things layer* e *platform layer*, conforme pode ser observado na Figura 3.5. Os componentes da camada *things layer* são utilizados para realizar o acesso aos provedores de dados, que podem ser sensores, pessoas ou plataformas de IoT que fornecem dados ao sistema. Na camada *platform layer* são agrupados os componentes responsáveis por permitir a programação dos fluxos de processamento utilizados na integração de serviços da IoT o que permitiria, por exemplo, construir um serviço de processamento contextual. Os principais componentes de cada camada são descritos a seguir:

Things layer

- ***Network Discovery and Registration Broker***: executa o gerenciamento de novos componentes agregados ao domínio, os quais, após conectados, recebem um identificador único;

- ***Low-Level Programming Libraries***: provê funções para acesso a dispositivos com diferentes protocolos de comunicação;

- ***API Mediator***: API criada em modelo *Representational State Transfer* (REST) que permite o acesso aos dispositivos e provedores de dados vinculados ao domínio em questão, estabelecendo um mecanismos de acesso unificado. A troca de informações é feita considerando o uso de linguagens de marcação como *JavaScript Object Notation* (JSON) e XML;

- ***API Access Management***: API REST que expõem os dispositivos descobertos e agregados para o domínio global da IoT. Os autores prevem o uso de conexões HTTPS e mecanismos de autenticação dos tipos OAuth [1] e OpenID [2].

Platform layer

- ***Thing Description Repository***: repositório constantemente atualizado pelas camadas de descoberta de recursos, que contém a descrição dos elementos agregados ao domínio;

- ***Two-Phase Discovery Engine***: este componente permite a descoberta de "coisas" que possuem APIs que podem ser utilizadas na composição de fluxos de processamento nas etapas posteriores de composição dos serviço em questão;

[1] http://oauth.net/
[2] http://openid.net/

- ***Flow Design***: disponibiliza uma interface para que o usuário possa encontrar e utilizar APIs disponíveis no domínio gerenciado. Esse componente considera também a interação humana através de dispositivos inteligentes, permitindo o acesso a APIs de comunicação para email e SMS, por exemplo;

- ***Flow Composition***: permite a combinação de dois ou mais fluxos para formar novos fluxos de processamento, gerando funcionalidades novas no sistema;

- ***Flow Execution Engine***: ambiente de execução e teste dos fluxos de processamento criados no sistema;

- ***Flow Template Management and Repository***: repositório onde são armazenados e gerenciados fluxos de processamento previamente definidos, permitindo a sua reutilização na composição de novos fluxos de processamento;

- ***Request Management***: componente que possibilita o acesso a fluxos de processamento. Caso o fluxo solicitado pelo usuário não seja encontrado no sistema, é exibida uma interface que permite a composição de um novo fluxo.

Para a composição dos fluxos de processamento, os autores utilizaram a ferramenta Anypoint Studio[3], a qual permite criar fluxos de processamento como o do exemplo mostrado na Figura 3.6, no qual é demostrado um fluxo que seria executado para encontrar um usuário capaz de tratar uma anomalia em um dado obtido no sistema.

[3]https://www.mulesoft.com/platform/studio

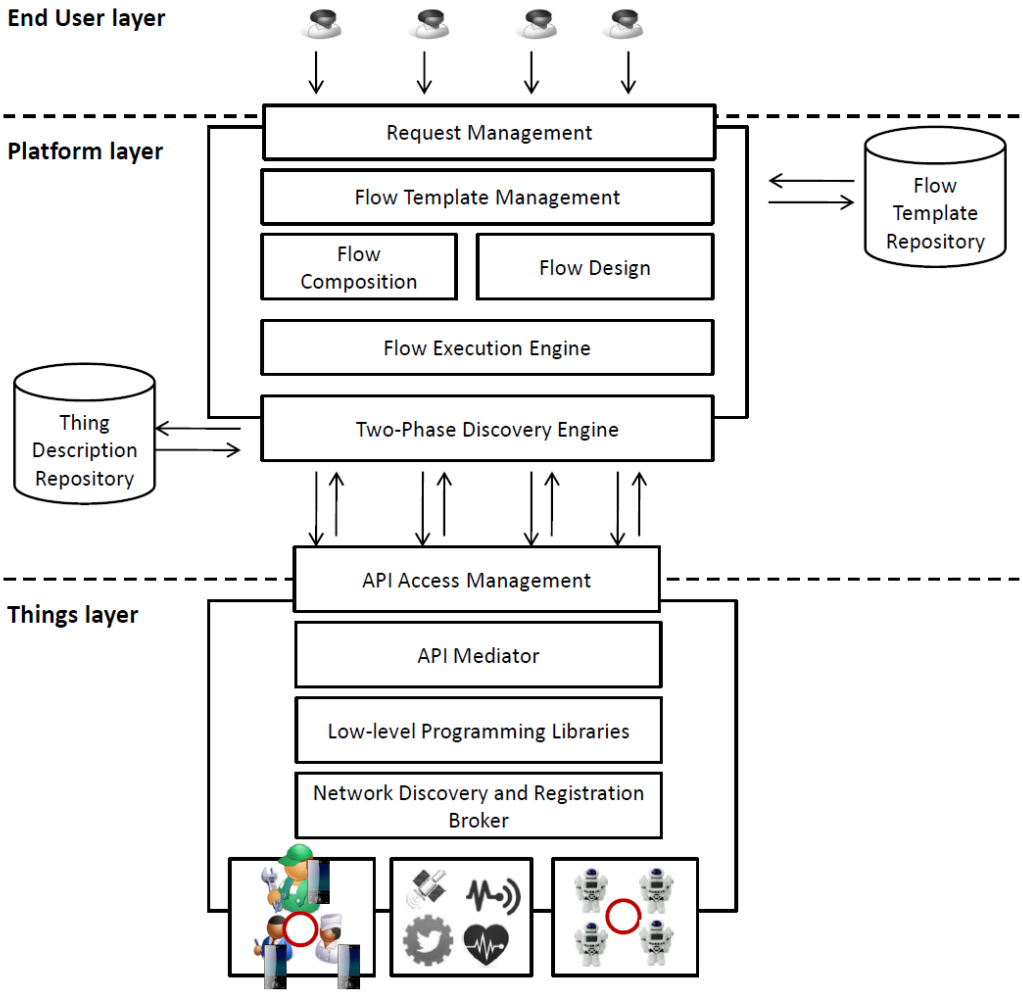

Figura 3.5 – Arquitetura proposta por Khodadadi, Dastjerdi and Buyya (2015).
Fonte: (KHODADADI; DASTJERDI; BUYYA, 2015).

3.4 Projeto CoCaMAAL

O *middleware* CoCaMALL – *A cloud-oriented context-aware middleware in ambient assisted living* (FORKAN; KHALIL; TARI, 2014) é baseado em tecnologia de computação em nuvem com arquitetura orientada por serviços – *Service-Oriented Architecture* (SOA). O mesmo tem como principal objetivo oferecer serviços de processamento contextual para ambientes de vivência assistida, abrangendo as etapas de coleta de dados de sensores, processamento e distribuição de dados de contexto.

Na Figura 3.7 pode ser vista a arquitetura genérica proposta no CoCaMAAL, a qual possui cinco componentes principais baseados em computação em nuvem. São eles:

- ***ALL Systems***: sistemas para ambientes de vivência assistida – *Ambient Assisted Living* (AAL), que representa os ambientes monitorados através do *middleware*, do qual são coletados dados de sensores do meio e também de redes de sensores vestí-

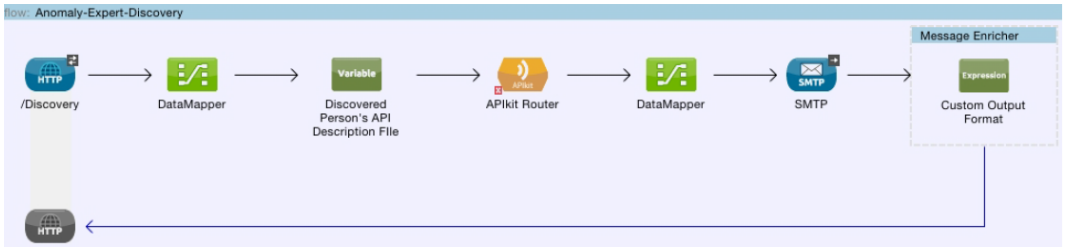

Figura 3.6 – Exemplo de fluxo de processamento criado *framework* proposto por Khodadadi, Dastjerdi and Buyya (2015).

Fonte: (KHODADADI; DASTJERDI; BUYYA, 2015).

veis – *Body Sensor Network*s (BSNs), os quais são enviados para os sistema de processamento contextual. Vários ambientes podem ser monitorados simultaneamente pelo *middleware*. Estes sistemas podem consumir dados de contexto gerados pelo *middleware* e também receber comandos de atuação;

- ***Context Aggregator and Providers (CAP)***: parte do sistema onde é feita a abstração das informações contextuais. Os dados de sensores são processados e convertidos para representações de alto nível interpretáveis por outros componentes da arquitetura. Para realizar estas tarefas os autores preveem o uso de técnicas de raciocínio e fusão de dados;

- ***Service Providers (SP)***: serviços e aplicações ligadas a ciência de situação. Podem ser aplicações que geram notificações e alertas, ou ainda cuidadores acionados em situações emergenciais;

- ***Context-aware middleware (CaM)***: o CaM é a parte mais importante da arquitetura, segundo os autores. No mesmo são feitas as etapas de processamento contextual, armazenamento, gerenciamento de serviços, controle de acesso e deliberação de ações de assistência;

- ***Context data visualization***: conjunto de interfaces gráficas para visualização de dados de pacientes por profissionais da área médica.

Dado um ambiente de vivência assistida, o fluxo de dados previsto para o *middleware* CoCaMALL pode ser visto na Figura 3.8. Exceto o ambiente monitorando, todos os componentes representados na figura são elementos de computação em nuvem. Os dados coletados dos sensores do ambiente e de BSNs são enviados para um gateway, o qual envia os dados para os sistemas de processamento contextual. Através das situações identificadas no processamento contextual, podem ser geradas ações de assistência, adaptação ou emissão de alertas.

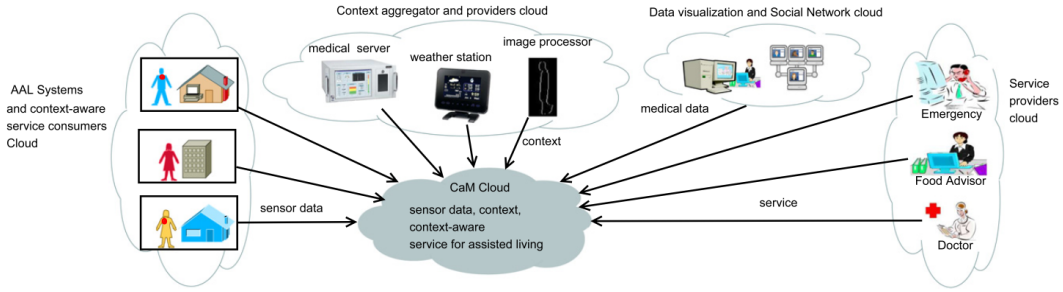

Figura 3.7 – Visão geral da arquitetura proposta por Forkan, Khalil and Tari (2014).
Fonte: (FORKAN; KHALIL; TARI, 2014).

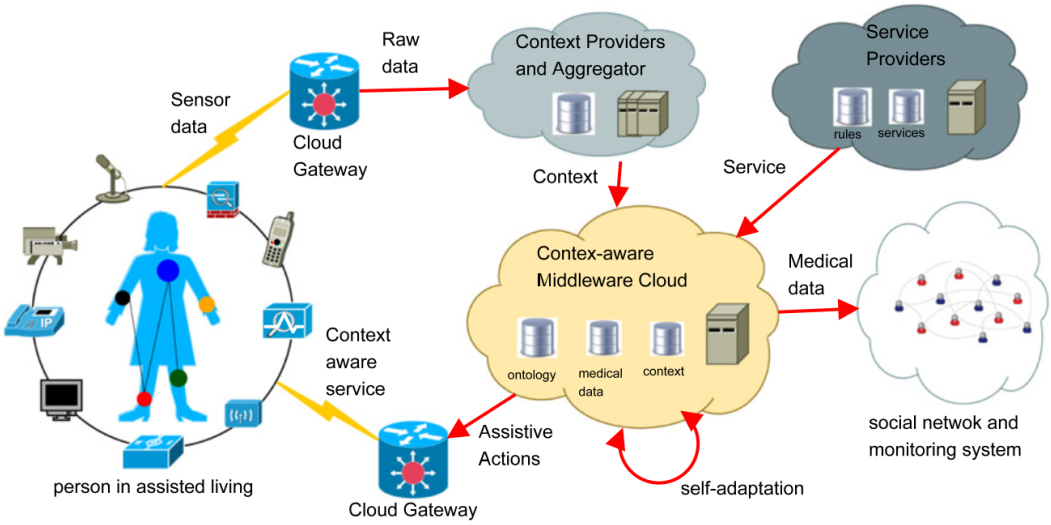

Figura 3.8 – Visão de fluxo de dados para um ambiente de vivência assistida gerenciado através do *middleware* CoCaMAAL.
Fonte: (FORKAN; KHALIL; TARI, 2014).

O uso de classificadores é uma técnica bastante difundida para o processamento contextual aplicado em dados de BSNs. Dentre estes classificadores, podem ser citados: *Artificial Neural Network* (ANN), Naive Bayes, K-means clustering, *Hidden Markov Model* (HMM), C.5 Decision Trees (BETTINI et al., 2010; RAVEENDRANATHAN et al., 2012). No *middleware* CoCaMAAL o fluxo de processamento contextual de um classificador é modelado de forma genérica conforme Figura 3.9.

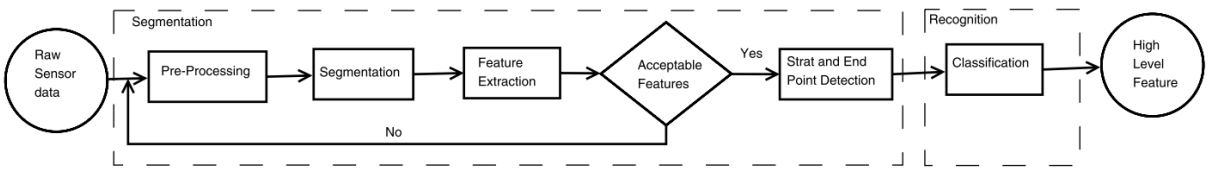

Figura 3.9 – Fluxo de processamento contextual para um classificador utilizando o *middleware* CoCaMAAL.
Fonte: (FORKAN; KHALIL; TARI, 2014).

3.5 Discussão dos Trabalhos Relacionados

Tendo por base as características e funcionalidades dos trabalhos relacionados, nesta seção é feita uma discussão acerca dos mesmos, considerando as principais premissas para concepção do EXEHDA-IS, listadas a seguir:

- (a) suporte para a criação e gerenciamento de fluxos de processamento contextual;
- (b) modelo híbrido para processamento de contexto na identificação de situações;
- (c) suporte à interação com sistemas e serviços da IoT;
- (d) possibilidade de uso em diferentes domínios de aplicação.

Na Tabela 3.1 é apresentada a comparação dos trabalhos, onde o símbolo "+" indica que o requisito é completamente atendido, o símbolo "-" denota que o requisito não é atendido e o símbolo "+/-" indica que o requisito é parcialmente atendido.

Tabela 3.1 – Comparação dos trabalhos relacionados.

	CARA	Programmable Context Awareness Framework	Simurgh	CoCaMAAL
a	-	+	+	+/-
b	+/-	-	-	+
c	+/-	+	+	+
d	-	+	+	-

Conforme os dados apresentados na Tabela 3.1, nota-se que nenhuma das plataformas analisadas atende a todos os requisitos considerados, sendo que a abordagem híbrida para processamento contextual foi atendida plenamente somente pelo *middleware* CoCaMAAL. Outro aspecto importante evidenciado é que as plataformas independentes de domínio de aplicação não oferecem suporte para o processamento híbrido de contexto, mesmo esta característica sendo relevante para diversos cenários de uso.

Uma característica comum nos trabalhos avaliados, e também em outros trabalhos como Giang et al. (2015), Heo et al. (2015), Azzara et al. (2014), Pérez et al. (2014), Blackstock and Lea (2014), é o uso de modelos *dataflow* (SOUSA, 2012) para processamento contextual.

Outro aspecto evidenciado nos trabalhos é a utilização de APIs do tipo REST. Dentre as vantagens do uso do modelo arquitetural REST podem ser citados: simplicidade de interpretação das APIs, desacoplamento dos serviços com sua representação e uso de padrões e protocolos abertos utilizados na web como o *Hypertext Transfer Protocol* (HTTP), JSON e XML (GUINARD; TRIFA; WILDE, 2010).

3.6 Considerações sobre o capítulo

Este capítulo apresentou plataformas que propõem infraestruturas para processamento contextual permitindo a criação de aplicações cientes de situação, considerando o uso de recursos da IoT. A descrição apresentada para cada projeto buscou caracterizar as arquiteturas e as estratégias de programação e processamento empregadas, avaliado ainda, se os trabalhos utilizam técnicas para processamento híbrido do contexto, suportam o gerenciamento de fluxos de processamento contextual, possuem suporte para IoT e apresentam arquitetura independente do domínio de aplicação.

Uma lacuna de pesquisa observada nos trabalhos estudados é a ausência de uma arquitetura que combine modelos de processamento híbrido de contexto com mecanismos para suporte a programação de fluxos de processamento contextuais. Nesse sentido, no próximo capítulo será apresentada a arquitetura do EXEHDA-IS, a qual busca através de um conjunto de componentes de softwares e interfaces, propor uma solução que combine estas duas funcionalidades.

4 EXEHDA-IS: VISÃO GERAL E FUNCIONALIDADES

Neste capítulo são apresentados os aspectos relacionados com a concepção da arquitetura do EXEHDA-IS, destacando as suas premissas e funcionalidades direcionadas para a identificação de situações na IoT.

4.1 Visão Geral

O objetivo do EXEHDA-IS é prover uma arquitetura de software que permita a **composição** e o **gerenciamento** de mecanismos para processamento contextual **híbrido** executado no servidor de contexto do *middleware* EXEHDA. Relacionado ao processamento contextual, entende-se nessa dissertação por:

- **composição** – criação de fluxos de processamento de dados de contexto que permita agregação de significado e elevação do nível de abstração dos dados possibilitando a identificação de situações;
- **gerenciamento** – representa o conjunto de atividades realizadas por usuários administradores sobre APIs do EXEHDA-IS, que permitem a criação, edição, execução, interrupção e exclusão de fluxos de processamento contextual;
- **híbrido** – processamento de contexto em que técnicas baseadas em especificação e aprendizado são combinadas.

Na perspectiva do *middleware* EXEHDA, esta dissertação busca contribuir com o seu Subsistema de Reconhecimento de Contexto e Adaptação. Nesse cenário, o desenvolvimento do EXEHDA-IS se deu considerando algumas premissas centrais, apresentadas a seguir:

- elaboração do modelo arquitetural considerando os componentes e técnicas já consolidadas no EXEHDA, permitindo a sua integração com este *middleware*;
- utilização de tecnologias e métodos entendidos como sendo o estado da arte na área de IoT e processamento contextual, evidenciados na revisão de literatura;
- criação de mecanismos que permitam a programação e o gerenciamento do processamento contextual em um nível de abstração que propicie fácil entendimento e utilização;

- utilização independente do domínio de aplicação. Essa característica deve ser atendida na abordagem proposta no EXEHDA-IS, uma vez que o *middleware* EXEHDA foi concebido para uso geral.

4.2 Arquitetura do Servidor de Contexto

O Servidor de Contexto é o elemento do EXEHDA que reune as funções de processamento e comunicação necessárias para composição do Subsistema de Reconhecimento de Contexto e Adaptação. No ambiente de IoT gerenciado pelo EXEHDA podem existir vários Servidores de Contexto, organizados em células, conforme mostrado na Figura 2.14.

O Servidor de Contexto é formado por cinco módulos: Aquisição, Atuação, Notificação, Comunicação e Processamento (LOPES et al., 2014b), os quais podem ser visualizados na Figura 4.1. Os elementos destacados na figura representam as partes onde estão as principais contribuições desta dissertação, que são: o Módulo de Processamento, o Módulo de Comunicação e a Interface de Configuração. Estes elementos e os demais componentes do Servidor de Contexto serão descritos a seguir.

Módulo de Aquisição: é responsável por prover suporte à captura das informações contextuais, coletadas pelos Servidores de Borda considerando sensores lógicos, implementados através de interfaces de software, e sensores físicos providos por interfaces de hardware.

Módulo de Atuação: controla a ativação, desativação e configuração dos atuadores, como consequência de uma notificação de outros módulos do Servidor de Contexto. Esse módulo recebe o identificador do atuador envolvido e os parâmetros operacionais a serem utilizados e interopera com os Servidores de Borda para disparo dos atuadores pertinentes.

Módulo de Processamento: tem como principal função realizar as tarefas pertinentes ao tratamento das informações contextuais, bem como dos eventos para identificar situações de interesse das aplicações. Para execução destas tarefas, é proposto nesta dissertação o uso de um modelo de processamento contextual híbrido, apresentado na Seção 4.3.

Módulo de Notificação: é responsável por notificar o resultado do processamento das informações de contexto e da consequente identificação de situações realizada pelo Módulo de Processamento. Esse módulo opera recebendo subscrições dos serviços e/ou aplicações que desejem notificações a respeito dos estados contextuais e das situações identificadas, interoperando através do Módulo de Comunicação.

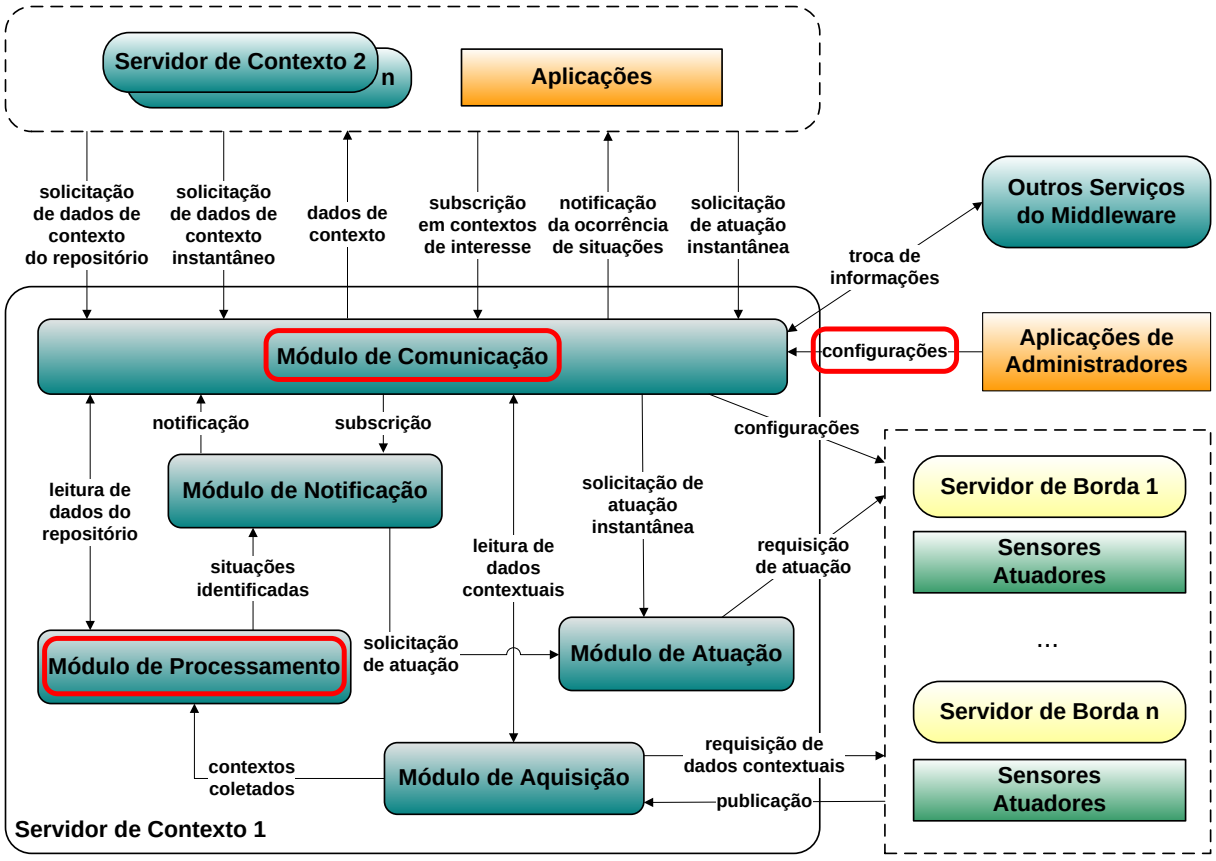

Figura 4.1 – Arquitetura do servidor de contexto.
Fonte: adaptado de (LOPES et al., 2014b).

Módulo de Comunicação: é utilizado por Servidores de Contexto remotos e/ou aplicações para solicitação de situações, dados contextuais e/ou o disparo de atuadores. Dispõe ainda de uma interface para configuração.

Interface de Configuração: permite que aplicações administradoras enviem configurações para o Servidor de Contexto. Estas configurações possibilitam a composição e o gerenciamento de fluxos de processamento contextual utilizando os mecanismos disponibilizados pelo Módulo de Processamento. A abordagem de programação proposta para esta interface é apresentada na Seção 4.4.

4.3 Arquitetura do Módulo de Processamento

A arquitetura do Módulo de Processamento proposto nesta dissertação foi elaborada visando o processamento híbrido de contexto. Desta forma, possui componentes que atendem a demandas relacionadas com técnicas de aprendizado, especificação e suporte para integração de ambas.

Técnicas baseadas em aprendizado permitem a análise de dados identificando padrões

e realizando o tratamento de incertezas. As técnicas baseadas em especificação, por sua vez, possibilitam a representação de situações através de modelos de raciocínio criados a partir do conhecimento de especialistas da área de aplicação. A combinação destes dois tipos de técnicas em um modelo híbrido permite que um número maior de situações possam ser identificadas, beneficiando-se do aprendizado do sistema e também do conhecimento de especialistas (LIM; VATS; CHAN, 2014; CHIBELUSHI, 2014; YE; DOBSON; MCKEEVER, 2012; BELLAVISTA et al., 2012).

O Processamento das informações contextuais para identificação de situações é feito utilizando três conjuntos de componentes e um conjunto de repositórios, mostrados no diagrama da Figura 4.2 e descritos a seguir.

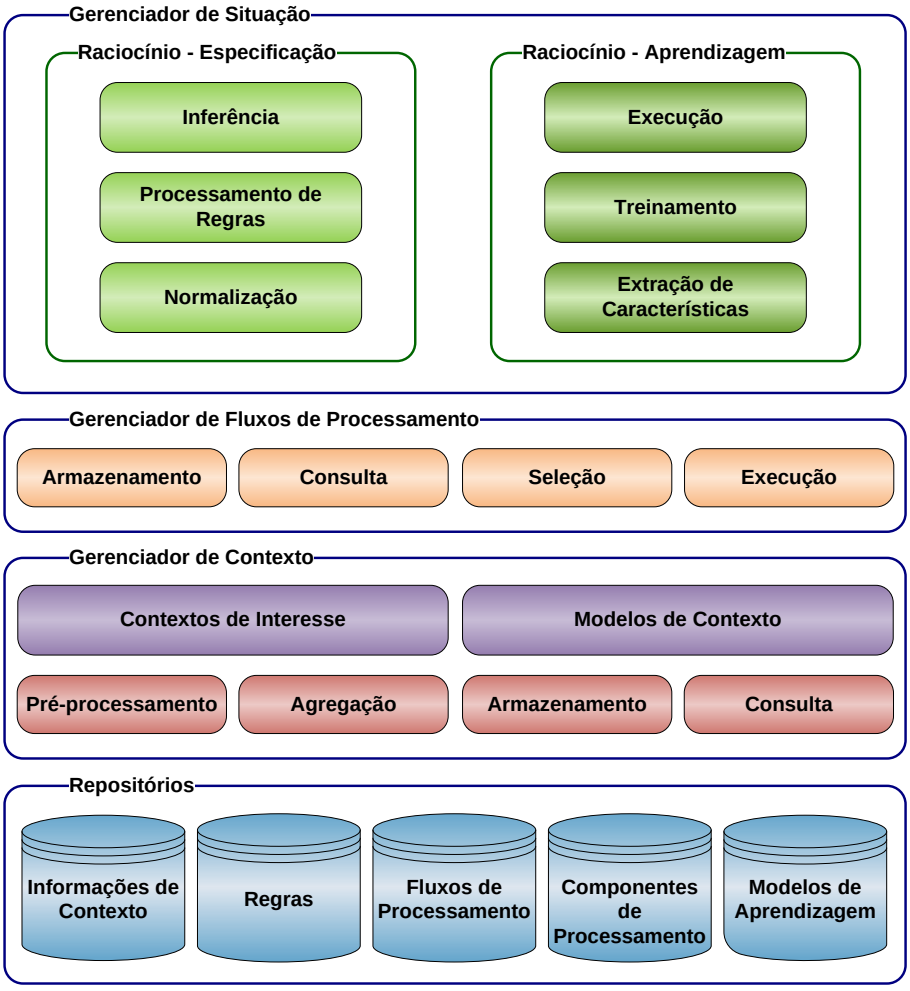

Figura 4.2 – Arquitetura do Módulo de Processamento do Servidor de Contexto do EXEHDA-IS.

Gerenciador de Contexto: é responsável por prover as funcionalidades necessárias para o processamento das informações contextuais, abrangendo as etapas de pré-processamento, agregação, armazenamento e consulta. Considerando os Contextos de Interesse das aplicações,

este processamento tem como objetivo elevar o grau de abstração dos dados de contexto coletados, melhorando a sua disponibilidade e usabilidade no processo de identificação de situações.

As informações manipuladas pelo gerenciador de contexto são armazenadas no Repositório de Informações de Contexto.

Gerenciador de Fluxos de Processamento: este gerenciador utiliza um mecanismo de eventos para disparar a execução dos fluxos de processamento. Estes eventos podem ser gerados pela alteração de dados de sensores ou outros contextos de interesse. A definição dos fluxos de processamento é armazenada no Repositório de Fluxos de Processamento através da API de Configuração, conforme descrito na Seção 4.4.

Gerenciador de Situações: disponibiliza as funcionalidades necessárias para o processo de raciocínio sobre os dados contextuais, visando a identificação de situações. Considerando a abordagem híbrida proposta nesta dissertação, os mecanismos de raciocínio foram divididos em duas categorias: Especificação e Aprendizagem.

O bloco de Aprendizagem contempla um dos desafios da abordagem híbrida proposta, que é permitir que os algoritmos de aprendizagem possam ser parametrizados e utilizados de maneira transparente para as aplicações, evitando desta forma que detalhes do seu funcionamento necessitem de tratamento por parte do desenvolvedor das aplicações.

Um dos recursos utilizados para permitir a abstração das técnicas de identificação de situações baseadas em aprendizagem é a inclusão de um mecanismo para Treinamento, o qual utiliza o mesmo modelo de dados independentemente do algoritmo escolhido. Dessa forma, o usuário fornece os dados de exemplo para o treinamento, escolhe a técnica que deseja utilizar e as regras para extração de características que devem ser executadas sob os dados de entrada. Após a execução do treinamento, os parâmetros obtidos são armazenados no repositório de Modelos de Aprendizagem. Estes parâmetros são acessados através das funções do mecanismo de Execução, utilizadas durante o processamento das aplicações.

Os algoritmos de processamento são armazenados na forma de componentes de software no Repositório de Componentes, sendo possível a inclusão de novos componentes. As interfaces dos componentes devem ser construídas de acordo com um modelo de interoperabilidade padronizado para o módulo em questão.

O mecanismo de Extração de Características é responsável por pré-processar as variáveis de contexto ou sinais lidos diretamente de sensores, extraindo o conjunto de características relevantes para o treinamento e posterior execução do algoritmo de aprendizagem. As rotinas para pré-processamento, assim como os algoritmos de aprendizagem, são componentes de software armazenados no Repositório de Componentes. Como exemplos de algoritmos para extração

de características, podem ser citados métodos estatísticos como média, variância, covariância, máximo e mínimo, ou ainda, métodos de processamento de sinais como FFT.

O bloco de Especificação permite que sejam empregadas técnicas baseadas em especificação para inferência de situações, considerando tanto os dados de contexto coletados, como os resultantes do processamento realizado no bloco de Aprendizagem. O mecanismo de Normalização transforma os valores numéricos destes dados contextuais para o domínio da técnica de especificação empregada. Por exemplo, no cenário de uso da Seção 5.2, é coletado o sinal vital de frequência cardíaca de pacientes, o qual é convertido para o domínio fuzzy através de funções de pertinência triangulares.

As regras relacionadas ao tipo de técnica de identificação de situação adotada são armazenadas no Repositório de Regras e através do mecanismo de Processamento de Regras são avaliadas e geram as informações necessárias para a identificação das situações realizada na etapa de Inferência.

A arquitetura do Módulo de Processamento do EXEHDA-IS, apresentada na Figura 4.2, é manipulada através de APIs e componentes de software que permitem compor os Fluxos de Processamento Contextuais, conforme descrito na Seção 4.4.

4.4 Abordagem para Programação do Processamento Contextual

O método para programação do processamento contextual proposto nesta dissertação considera o uso de um modelo *dataflow* (SOUSA, 2012), o qual foi observado como sendo uma tendência na área de processamento contextual, conforme análise da revisão de literatura apresentada na Seção 3.5.

Através do modelo de programação proposto, busca-se dar suporte à operação autônoma do EXEHDA-IS na identificação de situações. Isto é, uma vez programado o fluxo de processamento contextual que deve ser executado, o *middleware* realiza as operações de coleta de dados, processamento contextual e notifica as aplicações quando as situações de interesse são identificadas. A programação de fluxos de processamento faz-se necessária, visto que é nesta etapa que são definidos os requisitos das aplicações e a forma como os mecanismos do *middleware* serão alocados para atendê-los.

A estrutura de processamento foi modelada para permitir acesso através de uma API do tipo REST (FIELDING, 2000) para o Gerenciador de Fluxos de Processamento. Na Figura 4.3 pode ser vista a representação de um Componente de Processamento elementar da arquitetura. Para compor os Fluxos de Processamento Contextuais são criadas instâncias dos Componentes

de Processamento, as quais têm as suas entradas e saídas conectadas para obter o processamento desejado.

Figura 4.3 – Componente de Processamento elementar.

Como pode ser visto na Figura 4.3 os Componentes de Processamento podem receber como entrada uma mensagem, a qual, deve encapsular todas as variáveis necessárias para o processamento. Após o processamento podem ser geradas "N" mensagens de saída. Cada mensagem de saída pode encapsular uma estrutura de dados com "N" variáveis. No estudo de caso em Reabilitação Cardíaca, descrito na Seção 5.2, o encapsulamento dos dados das mensagens foi feito utilizando a linguagem de marcação JSON.

Na Figura 4.4 tem-se o modelo de processamento previsto para a composição dos Fluxos de Processamento Contextuais. A execução das Instâncias de Processamento é disparada por eventos de recepção de mensagens, as quais podem ser geradas na arquitetura do EXEHDA-IS por Fontes de Contexto ou por outros Fluxos de Processamento. Como Fontes de Contexto podem ser citados os servidores de Borda do EXEHDA.

Cada Fluxo de Processamento cadastro no repositório do EXEHDA-IS recebe um identificador (id), que permite a sua manipulação pelos mecanismos do Gerenciador de Fluxos de Processamento e também através da API de configuração. Cada fluxo de processamento deve possuir ao menos uma *Uniform Resource Identifier* (URI) definida na sua criação, que será utilizada para endereçar as mensagens para o fluxo de processamento.

As mensagens geradas internamente no Fluxo de Processamento são roteadas de forma automática de acordo com as conexões criadas no mesmo. Nos estudos de caso isto foi feito utilizando a ferramenta Node-RED, descrita na Seção 5.1, como solução para criação do Fluxo de Processamento e também para a sua execução.

Como pode ser visto na Figura 4.4, as terminações dos Fluxos de Processamento acionam APIs do EXEHDA-IS, representadas também por Componentes de Processamento. Na Figura 4.4 são referenciadas as APIs do Repositório de Informações de Contexto e do Gerenciados de Fluxos de Processamento, a quais permitem manipular informações de contexto e enviar mensagens para outros Fluxos, respectivamente. Pode ser visto também o componente genérico "API EXEHDA-IS", que permite configurar a URI de uma API do *middleware* para a qual será enviada a mensagem.

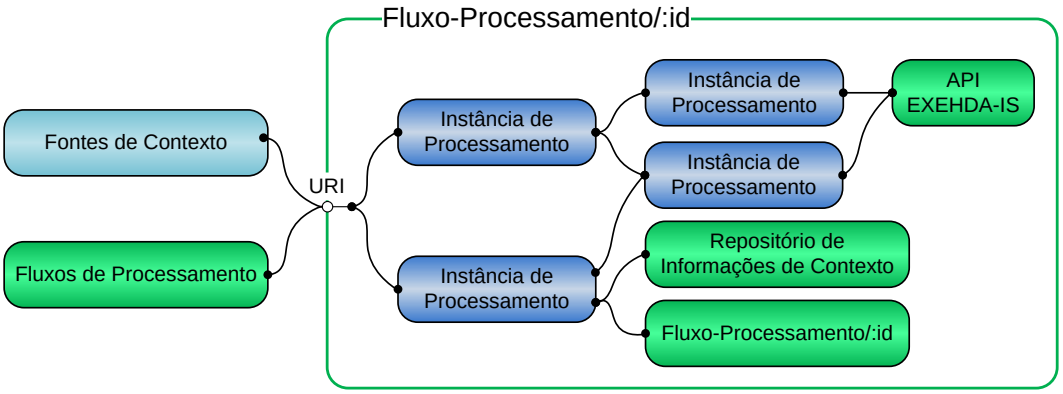

Figura 4.4 – Modelo de processamento previsto para a composição dos Fluxos de Processamento Contextual.

4.5 Considerações Sobre o Capítulo

Neste capítulo foi apresentada a visão geral da arquitetura do EXEHDA-IS, destacando as suas principais premissas e características. Foram apresentados também detalhes da arquitetura do Servidor de Contexto do EXEHDA, apontando os elementos em que se concentram as contribuições desta dissertação.

A arquitetura proposta foi apresentada em dois momentos. Primeiramente, foram destacados os componentes para suporte ao processamento contextual híbrido e a sua organização arquitetural que permite o gerenciamento de situações. Em um segundo momento, foi apresentada a abordagem *dataflow* proposta para a composição dos fluxos de processamento contextuais.

5 EXEHDA-IS: ESTUDO DE CASO E TECNOLOGIAS

Neste capítulo são apresentados os estudos de caso e as tecnologias de prototipação utilizadas para avaliação do EXEHDA-IS. O primeiro estudo de caso está relacionado com o monitoramento remoto da situação de saúde de pacientes em reabilitação cardíaca. O segundo é utilizado no gerenciamento de recursos em ambientes hospitalares, visando a redução de desperdícios.

5.1 Principais Tecnologias de Software Utilizadas

Na prototipação da arquitetura do EXEHDA-IS foram utilizadas algumas tecnologias de software para permitir o processamento dos dados, armazenamento e criação das interfaces de comunicação, descritas a seguir.

A ferramenta **Node-RED** foi utilizada como estratégia para edição gráfica dos Fluxos de Processamento Contextuais e também no servidor de contexto para criação do mecanismo de Execução dos Fluxos de Processamento. Foram combinados componentes de software nativos da ferramenta com componentes desenvolvidos para encapsular APIs do EXEHDA-IS e criar novas funções de processamento, conforme será detalhado nos estudos de caso. O Node-RED é uma ferramenta baseada em interface web, criada para conexão de dispositivos e API voltadas para a área de IoT. É implementada na linguagem JavaScript, utilizando o *framework* Node.js. Possui suporte nativo a JavaScript na ferramenta de edição gráfica e também no seu módulo para execução dos fluxos de processamento em servidores (BLACKSTOCK; LEA, 2014).

A **Linguagem Python** foi utilizada para a criação das camadas de interoperação dos componentes de processamento. Para criação de regras fuzzy foram utilizados os pacotes sk-fuzzy [1] e NumPy [2]. Python é uma linguagem de alto nível, orientada a objetos, de tipagem dinâmica, interpretada e interativa (BORGES, 2010).

A **Linguagem de Marcação JSON** foi utilizada em conjunto com a **Linguagem de Programação JavaScript** para formatação dos dados e criação de Componentes de Processamento utilizados nos Fluxos de Processamento Contextuais. A linguagem JSON é voltada para a troca de informações estruturadas entre linguagens de programação. Permite a representação textual de dados numéricos, listas e vetores (ECMA-404, 2013). O **Protocolo HTTP** foi utilizado como mecanismo de transporte dos dados entre Fluxos de Processamento e também para

[1] http://pythonhosted.org/scikit-fuzzy/
[2] http://www.numpy.org/

criação das APIs do EXEHDA-IS.

O Banco de Dados **MongoDB** foi utilizado para criação dos repositórios do EXEHDA-IS. O MongoDB é um tipo de banco de dados NoSQL baseado em documentos, de código aberto, projetado para alta performance e escrito em linguagem C++ (MONGODB, 2016). Os documentos são armazenados no formato JSON, esta característica facilita a integração com os fluxos de processamento do EXEHDA-IS, nos quais a linguagem JSON foi utilizada para comunicação de dados entre Componentes e Fluxos de Processamento.

O pacote de software **WekA** foi utilizado como solução para a criação dos componentes baseados em algoritmos de aprendizado de máquina. Este software reúne uma série de métodos utilizados na área de IA. O software permite a manipulação dos dados através de uma interface gráfica ou através de um conjunto de classes Java (WITTEN; FRANK; HALL, 2011). As classes em Java foram a solução utilizada no EXEHDA-IS. Os dados utilizados no treinamento e processamento dos algoritmos de aprendizado foram representados no formato ARFF.

5.2 Estudo de Caso 1: Ciência de Situação na Reabilitação Cardíaca

A reabilitação cardíaca pode envolver diversas terapias, incluindo administração de medicamentos, aconselhamento nutricional e também a prescrição de atividades físicas. A reabilitação cardíaca através de atividades físicas é considerada uma terapia central. Estudos indicam que a reabilitação baseada em exercícios físicos foi associada a uma redução de 20 a 30% nas taxas de mortalidade, quando comparada com cuidados sem exercício (RABELO; GIL; ARAÚJO, 2006).

Um aspecto que deve ser considerado na terapia através de exercícios físicos diz respeito a respostas desproporcionais na frequência cardíaca. Isto pode indicar uma situação de risco para o paciente. Desta forma, o reconhecimento da atividade física e sua correlação com a frequência cardíaca pode permitir uma recuperação mais segura (NEGRÃO; BARRETO, 2010). Nesse sentido, o emprego de sistemas computacionais para detecção autônoma de atividades e a correlação destas com parâmetros fisiológicos pode minimizar a necessidade de intervenção do próprio paciente ou do terapeuta na identificação de situações de risco.

Como forma de avaliar as funcionalidades da arquitetura do EXEHDA-IS foi proposta uma aplicação para monitoramento de pacientes em reabilitação cardíaca após um acidente vascular. Essa aplicação explora as funcionalidades de processamento híbrido e também a Interface de Configuração do EXEHDA-IS.

A aplicação consiste em classificar a atividade física realizada pelo paciente, e correlacioná-

la com a frequência cardíaca, permitindo identificar situações de risco, caso os parâmetros estejam fora da faixa de normalidade estabelecida pelo terapeuta. A saída produzida representa o nível de risco de saúde do paciente, classificado no domínio linguístico: “baixo”, “moderado” e “alto”. Na Figura 5.1 pode ser visto o fluxo de coleta e processamento de dados contextuais proposto.

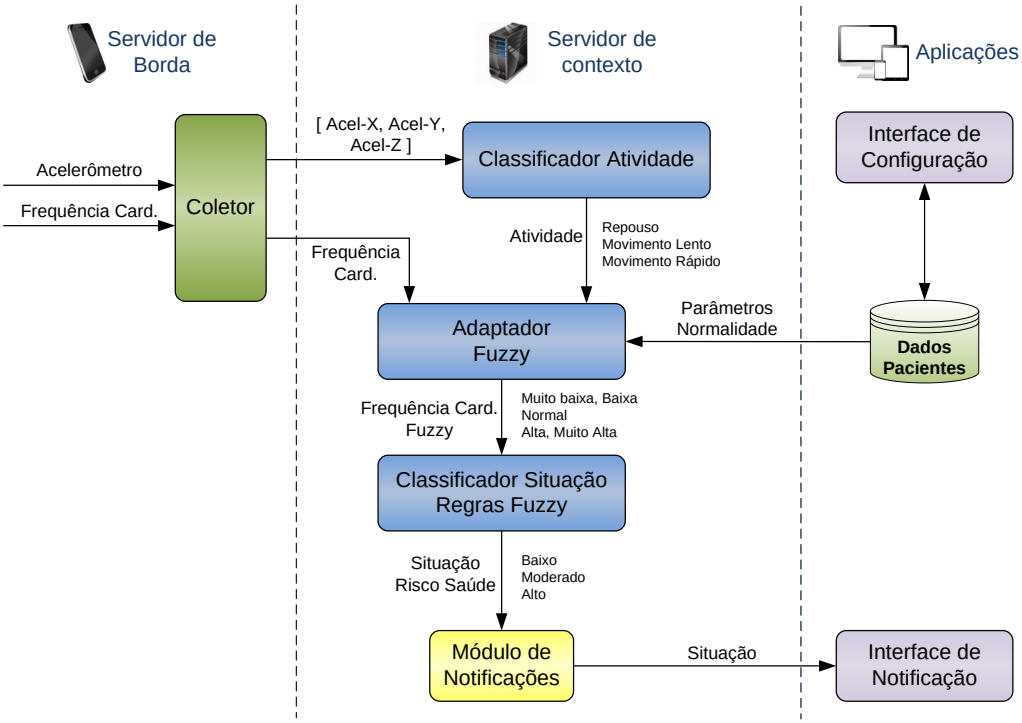

Figura 5.1 – Visão geral do estudo de caso proposto na área de reabilitação cardíaca.

A identificação da atividade física do paciente foi feita utilizando algoritmos de classificação baseados em técnicas de aprendizado executadas sobre os sinais coletados do *smartphone* do paciente. As regras para inferência do risco de saúde foram criadas utilizando lógica fuzzy, considerando a atividade física, a duração da atividade e a frequência cardíaca do paciente.

5.2.1 Identificação de Atividades

O treinamento e o teste do componente para a classificação de atividades foi feito utilizando a base de dados disponibilizada no trabalho de Kwapisz, Weiss and Moore (2011), no qual foram capturados sinais de acelerômetros de *smartphones* de 29 voluntários durante a execução das seguintes atividades físicas: caminhando, correndo, subindo escada, descendo escada, sentado e de pé. Os autores disponibilizaram os sinais pré-processados com as características extraídas e também os valores lidos diretamente do acelerômetro do celular.

Tabela 5.1 – Percentual de atividades classificadas corretamente utilizando as características originais do estudo de Kwapisz, Weiss and Moore (2011).

	Árvore de Decisão	Rede Bayesiana	Rede Neural	Regressão Lógica
Caminhando	87.33%	82.94%	89.23%	92.37%
Correndo	95.80%	94.19%	98.89%	97.41%
Subindo escadas	59.21%	39.31%	52.96%	24.51%
Descendo escadas	54.66%	30.64%	40.17%	10.35%
Sentado	98.66%	94.98%	98.33%	95.32%
De pé	97.56%	84.96%	93.90%	91.06%
Total	84.88%	77.50%	84.39%	79.05%

Tabela 5.2 – Percentual de atividades classificadas corretamente utilizando o novo conjunto de características.

	Árvore de Decisão	Rede Bayesiana	Rede Neural	Regressão Lógica
Caminhando	92,31%	92,55%	93,35%	91,37%
Correndo	97,41%	98,71%	97,35%	96,94%
Subindo escadas	76,81%	60,50%	75,63%	63,03%
Descendo escadas	68,98%	62,86%	51,22%	40,00%
Sentado	97,90%	94,41%	93,71%	94,06%
De pé	97,40%	99,57%	96,54%	94,81%
Total	90,61%	88,68%	89,01%	85,65%

Foram feitos testes com quatro tipos de classificadores: Árvore de Decisão C4.5, Rede Bayesiana, Rede Neural e Regressão Lógica. Estes classificadores são apontados na literatura como os mais utilizados na identificação de atividades através de sensores vestíveis (LARA; LABRADOR, 2012).

O primeiro teste realizado consistiu em executar a classificação de atividades sobre as características já pré-processadas por Kwapisz, Weiss and Moore (2011). Os resultados para este teste podem ser vistos na Tabela 5.1. Os autores utilizaram um total de 43 características do sinal, incluindo: média, desvio padrão e tempo entre picos, além de um método empírico de distribuição intervalar de máximos e mínimos que gera 30 características. O último método não foi claramente especificado pelos autores.

O segundo teste realizado consistiu em treinar os classificadores utilizando um novo conjunto de características extraídas dos sinais lidos do acelerômetro. O sinal do acelerômetro foi obtido com uma taxa de 20 amostras por segundo e o processamento das características executado em janelas temporais de 10 segundos, gerando um conjunto com 15 características, listadas a seguir:

- frequência das três componentes de maior amplitude da FFT calculada sobre o módulo da aceleração resultante;
- aceleração média nos eixos x, y e z;
- variância nos eixos x, y e z;
- aceleração máxima nos eixos x, y e z;
- aceleração mínima nos eixos x, y e z.

O resultado obtido com o novo conjunto de características pode ser visto na Tabela 5.2. Percebe-se que o índice de classificação correta aumentou em cerca de 7,0%. Além disso, o número de características utilizadas foi reduzido de 43 para 15. Isso simplifica o treinamento e a execução dos modelos de aprendizado.

Com a finalidade de adequar as atividades para a aplicação proposta e também de melhorar o índice classificação correta, as atividades foram agrupadas como:

- **Repouso**: sentado e de pé;
- **Movimento Lento**: caminhando, descendo escada e subindo escada;
- **Movimento Rápido**: correndo.

Na Figura 5.2 pode ser visto o histograma das componentes de frequência com maior amplitude para as atividades classificadas. Nota-se no mesmo que existe uma diferença bem definida da frequência média para cada atividade. Isso indica que a FFT é uma boa característica para uso nos classificadores.

Utilizando a classificação agrupada e o conjunto de características proposto nesta dissertação, o processo de treinamento e teste dos classificadores foi repetido. O classificador Árvore de Decisão se manteve com o melhor índice de acerto. Após o treinamento, foi gerada uma Árvore de Decisão com 40 ramos e 79 elementos. O resultado da classificação de atividades para a Árvore de Decisão é apresentado na Tabela 5.3. O percentual total de classificação correta obtido foi de **98,32%**.

Para realizar as etapas de extração de características, armazenamento de exemplos, treinamento e classificação de atividade a partir do sinal obtido do acelerômetro do *smartphone*, foi criado no EXEHDA-IS o fluxo de processamento mostrado na Figura 5.3.

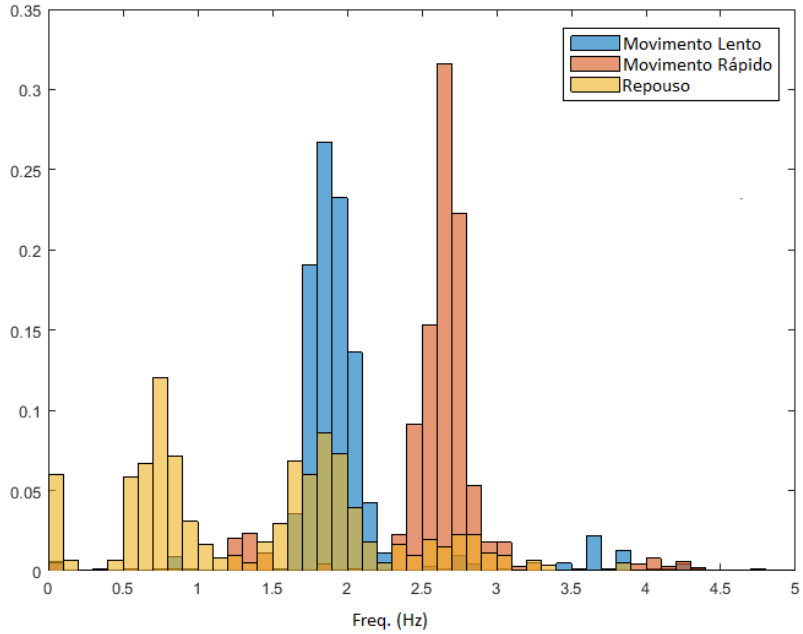

Figura 5.2 – Histograma da distribuição em frequência para as componentes de maior amplitude para cada atividade classificada.

Tabela 5.3 – Matriz de confusão para classificação de atividades utilizando Árvore de Decisão.

	Movimento lento	Movimento rápido	Repouso
Movimento lento	**98,60%**	2,36%	1,15%
Movimento rápido	1,25%	**97,64%**	0,00%
Repouso	0,16%	0,00%	**98,85%**
Total correto	**98,32%**		

O Fluxo possui três URIs: (I) "*/activity-example*" – permite extrair as características do sinal e armazenar um exemplo para posterior treinamento do classificador baseado em aprendizado; (II) "*activity-train*" – realiza o treinamento de um classificador para atividades utilizando a base de exemplos armazenado no Repositório de Modelos de Aprendizagem; (III) "*activity-classify*" – executa a extração de característica e classificação da atividade, enviando o resultado para o repositório de informações de contexto.

Os componentes "*classify*" e "train" encapsulam o acesso às APIs de Execução e Treinamento do módulo Raciocínio-Aprendizagem da arquitetura do EXEHDA-IS. Na etapa de extração de características são utilizados os componentes *fft, mean, max* e *variance*, criados especificamente para este estudo de caso. Os mesmos são descritos a seguir:

- ***fft***: recebe um vetor de amostras e a frequência de amostragem. Retorna um vetor com a magnitude da FFT calculada e outro vetor com as frequências correspondentes a cada ponto do vetor de magnitude;

- ***mean***: recebe um vetor de amostras e retorna o valor médio das amostras;

- ***max***: recebe um vetor de amostras e a configuração de "N" valores máximos que devem ser retornados e retorna um vetor com os "N" maiores valores encontrados no vetor de amostras;
- ***variance***: recebe um vetor de amostras e retorna a variância das amostras.

O componente "*format-msg*" executa a formatação da mensagem recebida para o padrão esperado pelos componentes subsequentes do fluxo. No bloco "*join*" as mensagens assíncronas geradas pelos componentes antecedentes são agrupadas em uma única mensagem. O bloco "*switch*" realiza o direcionamento da mensagem resultante para o Repositório de Modelos de Aprendizagem ou para o classificador de atividades de acordo com a URI que foi acionada. O valor resultante da classificação é armazenado no Repositório de Informações de Contexto.

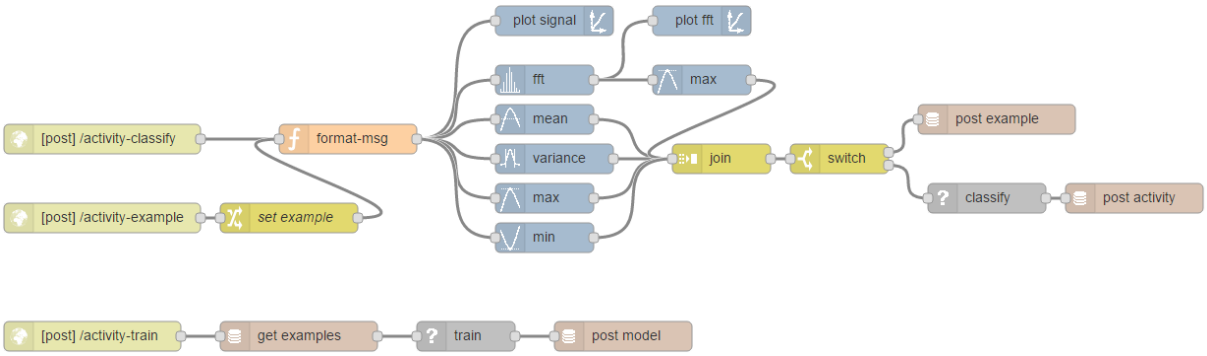

Figura 5.3 – Fluxo de processamento criado para executar a extração de características do sinal do obtido através do acelerômetro do *smartphone*.

Os componentes "*plot-signal*" e "*plot-fft*" permitem a visualização gráfica do último valor gerado nos ramos do fluxo em que estão conectados. Nas Figuras 5.4 e 5.5 podem ser vistos o sinal do acelerômetro e sua respectiva FFT, obtidos durante a extração de características de um exemplo para Movimento Lento.

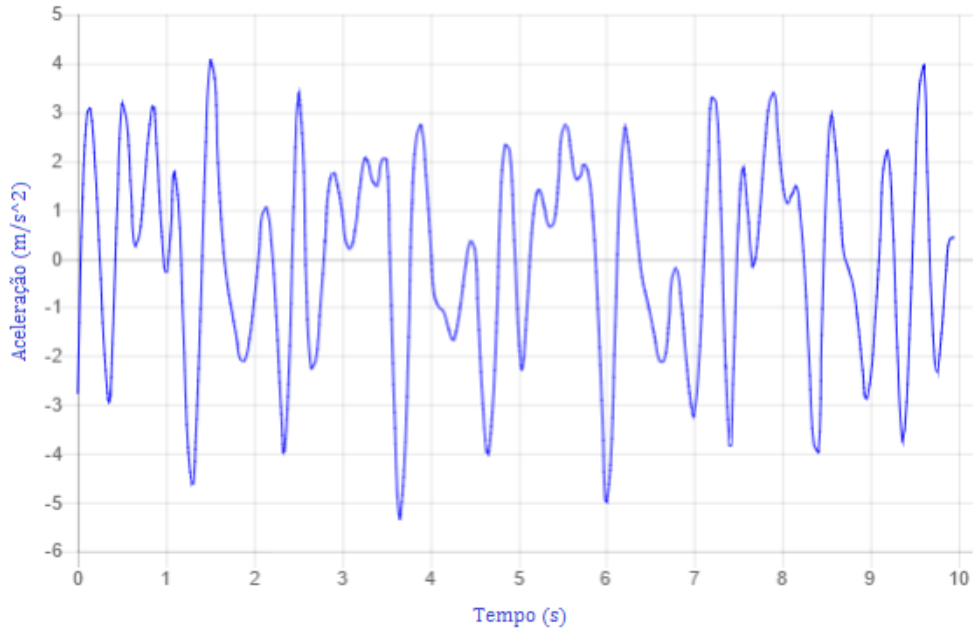

Figura 5.4 – Sinal obtido do acelerômetro durante a atividade Movimento Lento.

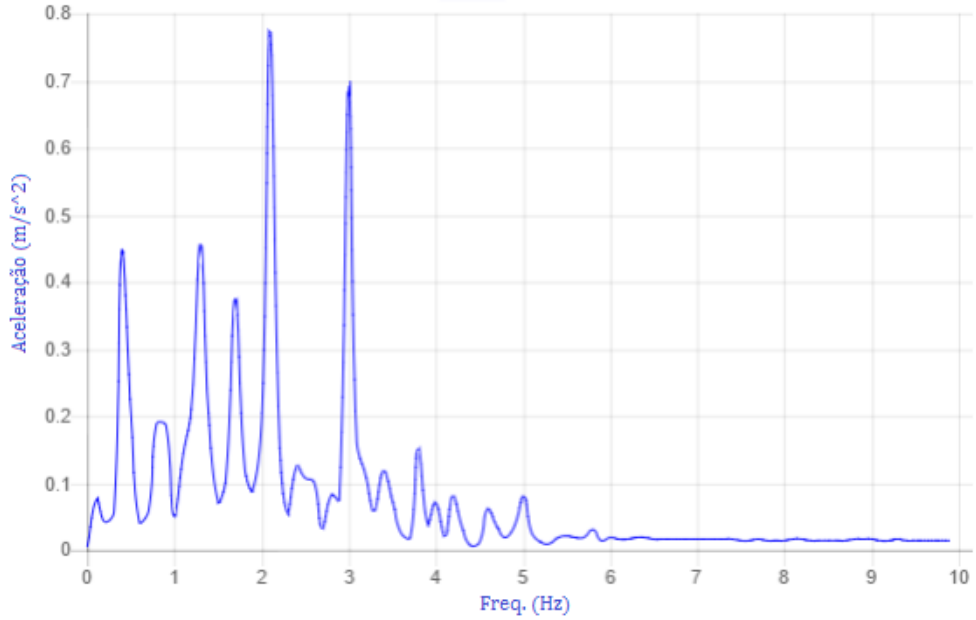

Figura 5.5 – FFT do sinal mostrado na Figura 5.4 obtida através do Fluxo de Processamento mostrado na Figura 5.3.

```
{
    "_id": "582e81f79cd38ddc81db02e5",
    "context": "activity-classifier",
    "var": "example",
    "pack_id": 1,
    "activity": "Movimento Lento",
    "max_fft": [
        2.1,
        3,
        1.3
    ],
    "mean": [
        0.06286572080391523,
        9.523722119283848,
        1.6976173724938122
    ],
    "variance": [
        4.267586039727258,
        18.295197526230705,
        8.822424127853983
    ],
    "max": [
        4.10097601405728,
        18.1974160178213,
        13.1772487008463
    ],
    "min": [
        -5.2012468410914,
        0.161246364081685,
        -3.82665130832256
    ]
}
```


Figura 5.6 – Dados incluídos no repositório de informações de contexto após a extração de características de um exemplo para treinamento.

Os exemplos para treinamento são armazenados no Repositório de Modelos de Aprendizagem criado através do banco de dados NoSQL MongoDB. Na Figura 5.6 pode ser visto um documento criado neste repositório contendo os dados de um exemplo para treinamento do classificador de atividade.

5.2.2 Inferência de Situação

Na etapa de inferência de situação é feita a determinação do Nível de Risco para a Saúde do paciente baseado na sua frequência cardíaca e na atividade física realizada, previamente classificada conforme descrito na Seção 5.2.1.

A frequência cardíaca apropriada para cada atividade física é obtida através de uma interface para a aplicação em que é feito o gerenciamento dos pacientes. O domínio linguístico para frequência cardíaca é dado por: “muito baixa”, “baixa”, “normal”, “alta” e “muito alta”. No bloco **Adaptador Fuzzy**, representado na Figura 5.1, os valores numéricos obtidos do sensor de batimento cardíaco são transformados para o domínio fuzzy aplicando funções de pertinência triangulares, parametrizadas com os padrões de normalidade do paciente. O mesmo procedimento é realizado para a variável que representa a duração da atividade, classificada como “Baixa”, “Média” e “Alta”.

A escolha da lógica fuzzy para a criação das regras de classificação de situações se deu tendo em vista os seus mecanismos voltados ao tratamento de dados imprecisos e construção de algoritmos através de modelos interpretáveis. Isso facilita a criação de regras por especialistas da área de aplicação (SOBREVILLA; MONTSENY, 2003).

A inferência foi feita aplicado o método Mamdani, no qual são previstos quatro módulos: *fuzzificação*, base de regras, inferência e *defuzzificação* (SHAW; SIMõES, 2007).

Na Figura 5.7 podem ser vistos exemplos de funções de pertinência utilizadas na *fuzzificação* da frequência de batimento cardíaco e duração da atividade, baseadas em valores médios apresentados por Negrão and Barreto (2010). Nota-se que são estabelecidas funções de pertinência para a frequência cardíaca em cada uma das atividades que podem ser identificadas pela aplicação. Isto simplifica a criação da base de regras fuzzy, visto que a frequência cardíaca tratada nas regras é previamente parametrizada pelo tipo de atividade.

A base de regras fuzzy criada para o estudo de caso pode ser vista na Figura 5.8. Foram aplicados três tipos de operadores fuzzy:

- OR = Máximo(x,y);
- AND = Mínimo(x,y);
- NOT = 1 - x.

Onde: x e y são valores de pertinência de variáveis linguísticas (SHAW; SIMõES, 2007).

Cada regra é formada por um conjunto de antecedentes que, relacionados através dos operadores fuzzy, geram um valor de pertinência resultante. Este é utilizado para ponderar a função de pertinência da variável linguística consequente. Neste trabalho foi aplicado o método *clipped*, onde a função de pertinência do consequente é “cortada” para o nível de pertinência resultante dos antecedentes avaliados na regra (SHAW; SIMõES, 2007).

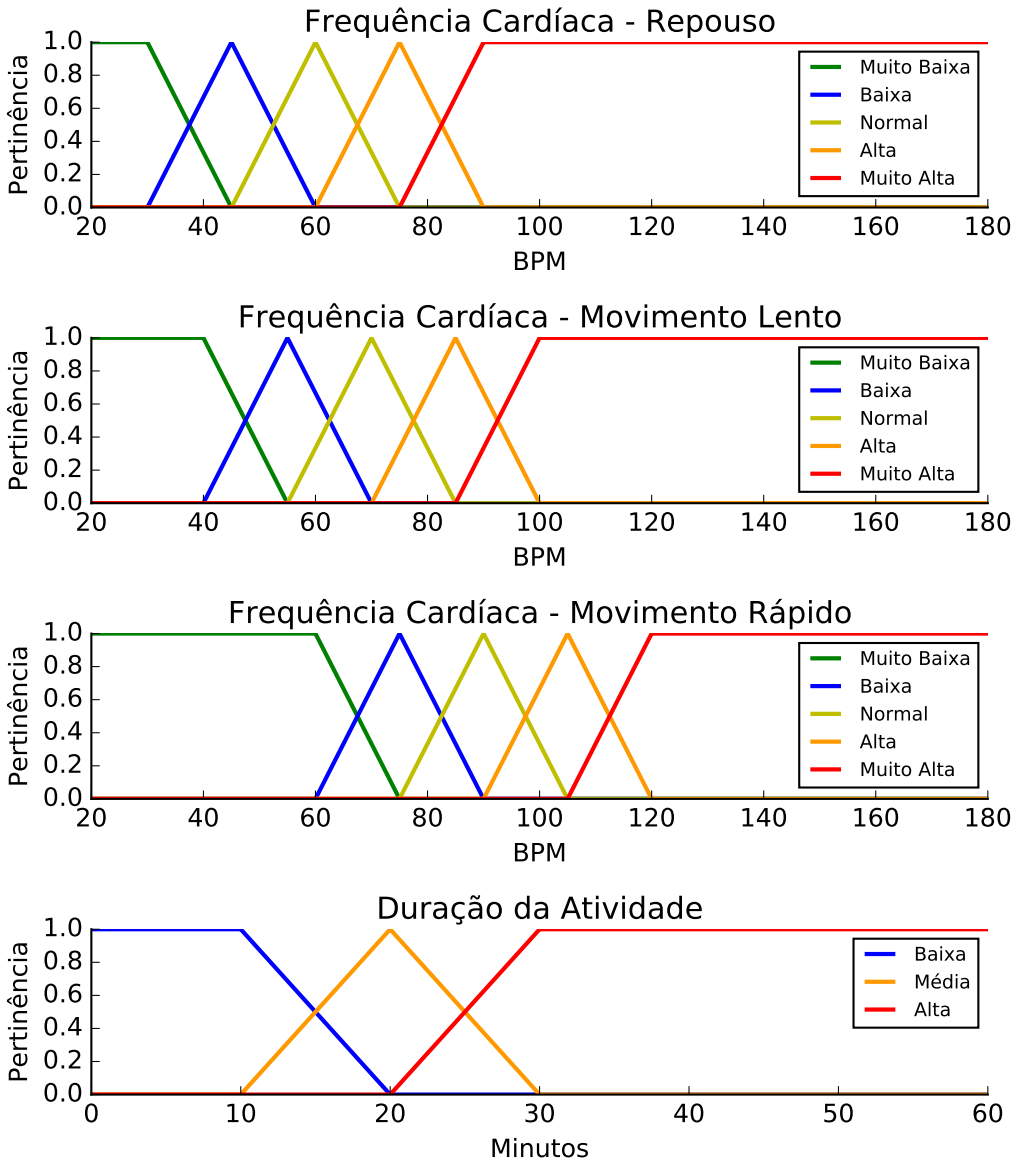

Figura 5.7 – Funções de pertinência para frequência cardíaca para os três tipos de atividades identificadas pela aplicação.

Após a execução das regras, são geradas funções de pertinência agregadas para cada variável linguística do conjunto de saída, as quais são agregadas novamente formando a função de pertinência que será aplicada na etapa de *deffuzificação*.

A saída agregada obtida através das regras foi *deffuzificada* utilizando o método do centroide (SHAW; SIMõES, 2007). Na Figura 5.9 são mostradas as funções de pertinência envolvidas no processo de inferência da situação de Risco para Saúde considerando a atividade Movimento Rápido com duração de 40 minutos e frequência cardíaca de 110 BPM. A situação inferida após a defuzzificação foi Risco Moderado.

```
IF freq_card IS normal THEN risco IS baixo
IF atividade IS repouso AND freq_card IS alta THEN risco IS moderado
IF atividade IS repouso AND freq_card IS baixa THEN risco IS moderado
IF atividade IS repouso AND freq_card IS muito_alta THEN risco IS alto
IF atividade IS repouso AND freq_card IS muito_baixa THEN risco IS alto
IF atividade IS NOT repouso AND duracao IS baixa OR media AND freq_card IS alta THEN risco IS moderado
IF atividade IS NOT repouso AND duracao IS baixa OR media AND freq_card IS baixa THEN risco IS moderado
IF atividade IS NOT repouso AND duracao IS baixa OR media AND freq_card IS muito_alta THEN risco IS alto
IF atividade IS NOT repouso AND duracao IS baixa OR media AND freq_card IS muito_baixa THEN risco IS alto
IF atividade IS NOT repouso AND duracao IS alta AND freq_card IS alta THEN risco IS alto
IF atividade IS NOT repouso AND duracao IS alta AND freq_card IS baixa THEN risco IS alto
IF atividade IS NOT repouso AND duracao IS alta AND freq_card IS muito_alta THEN risco IS alto
IF atividade IS NOT repouso AND duracao IS alta AND freq_card IS muito_baixa THEN risco IS alto
```

Figura 5.8 – Base de regras para inferência da situação de Risco para Saúde do paciente.

Para verificar o funcionamento da aplicação desenvolvida, foram gerados dados através de um *smartphone*, enviando para o EXEHDA-IS valores de frequência e cardíaca e também vetores de aceleração obtidos do banco de dados disponibilizado por Kwapisz, Weiss and Moore (2011), descrito na Seção 5.2.1.

O registro gráfico criado para uma sequência simulada nesse cenário pode ser visto na Figura 5.10. Na mesma, está destacado através do cursor vertical o momento em que foi inferida uma situação com risco Alto.

Na Figura 5.11 pode ser visto o e-mail enviado para o terapeuta, através da interface de notificações do EXEHDA-IS, após a detecção da situação de risco mostrada na figura 5.10.

5.2.3 Análise do Estudo de Caso

O estudo de caso em reabilitação cardíaca, apresentado nesta seção, permitiu demostrar o uso da arquitetura do EXEHDA-IS principalmente com relação a composição e execução de Fluxos de Processamento Contextuais Híbridos. Foram evidenciadas as etapas de extração de características e treinamento, fundamentais para a criação de elementos de processamento contextual baseados em aprendizagem. Por fim, o uso da lógica fuzzy como ferramenta para raciocínio baseado em especificação, permitiu demostrar o uso combinado de técnicas de aprendizagem e especificação, pressuposto dos modelos híbridos para identificação de situações.

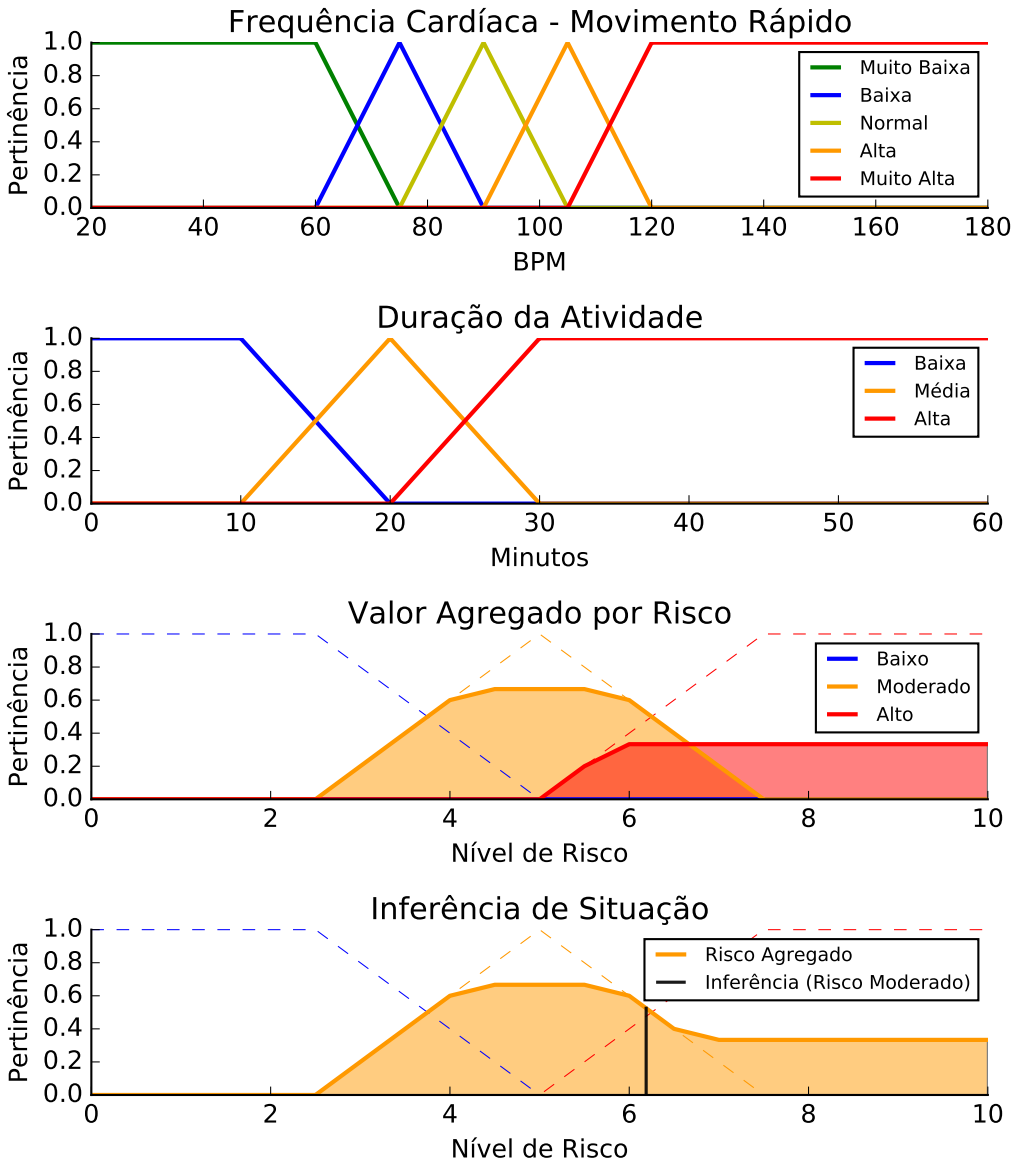

Figura 5.9 – Exemplo de inferência de situação de Risco para Saúde utilizado as regras em lógica fuzzy.

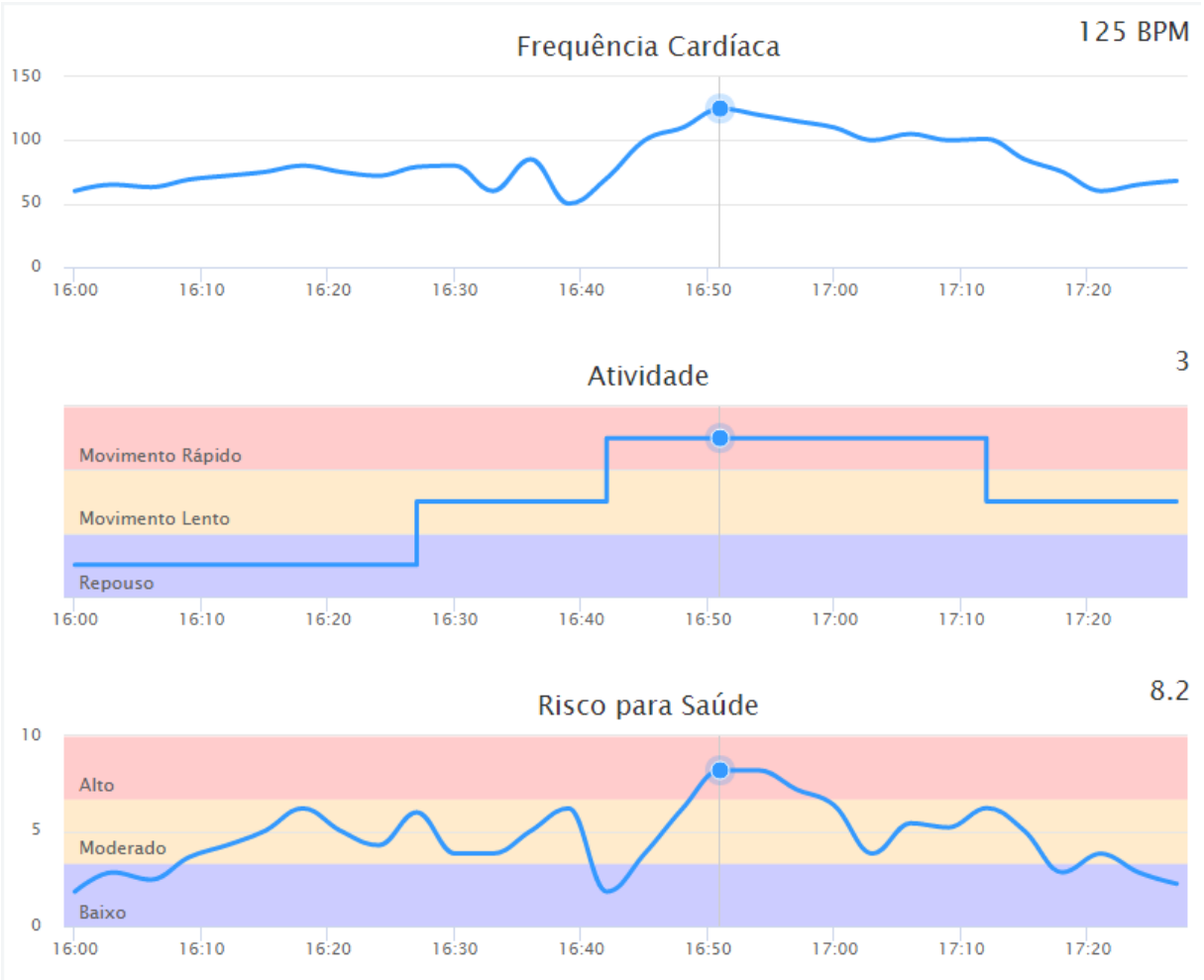

Figura 5.10 – Registro gráfico de frequência cardíaca, atividade e inferência de risco realizadas pelo EXEHDA-IS.

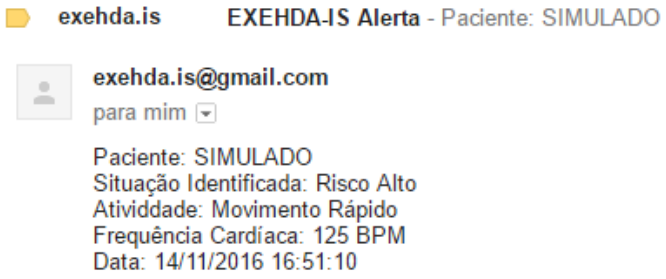

Figura 5.11 – E-mail enviado pelo EXEHDA-IS após a detecção de uma situação de Risco para a Saúde do paciente simulado.

5.3 Estudo de Caso 2: Ciência de Situação em Infraestrutura Hospitalar

O desperdício de recursos em uma infraestrutura hospitalar torna os serviços mais onerosos para a instituição e também pode prejudicar a qualidade do serviço prestado. Frente a este problema, a Universidade Católica de Pelotas (UCPel) tem o objetivo de implantar no Hospital Universitário São Francisco de Paula (HUSFP) um sistema que auxilie na identificação de situações de desperdício de recursos, visando minimizá-las.

Considerando essa iniciativa da UCPel foi concebido o estudo de caso descrito nesta seção, cuja finalidade principal é avaliar o EXEHDA-IS em regime de monitoramento de longo prazo associado a procedimentos de atuação remotos e acesso através de ambientes para gerenciamento e visualização de dados contextuais.

A especificação da aplicação em questão foi realizada em conjunto com o setor de Engenharia Clínica do HUSFP. A seguir serão descritos alguns requisitos determinados em análise conjunta:

- **medição de vazão**: medir o fluxo de óxido nitroso para vazões de até 300 l/min em uma tubulação com pressão de até 400 kPa;
- **medição de temperatura**: medir a temperatura do ambiente onde está alojado o cilindro de óxido nitroso na faixa de -5 ^{o}C até 50 ^{o}C;
- **painel de comando e alertas**: disponibilizar um painel instalado próximo ao cilindro de óxido nitroso, que permita sinalizar a troca do mesmo e que, através de indicadores luminosos informe a estimativa de volume do cilindro;
- **estimativa de volume**: baseado na medição de vazão realizar o cálculo de volume de óxido nitroso disponível no cilindro;
- **projeção de reabastecimento**: baseado na estimativa de volume atual e na previsão de demanda do hospital, realizar o cálculo da data prevista para substituição do cilindro;
- **identificação de situações de desperdício**: baseado na demanda de óxido nitroso prevista para um determinado momento realizar a identificação de possíveis situações de desperdício, caso a vazão medida esteja fora da faixa esperada.

O emprego da arquitetura do EXEHDA para este estudo de caso está representado na Figura 5.12. Em cada unidade monitorada é instalado um Servidor de Borda do EXEHDA, o

qual faz a coleta de dados dos sensores, e os envia para o Servidor de Contexto do EXEHDA-IS. O Servidor de Contexto recebe também as informações de demanda prevista para as unidades onde o óxido nitroso é utilizado. Isso é feito através de uma interface para o sistema de gerenciamento do hospital.

As informações do ambiente são coletadas e armazenadas em regime continuado. Isso permite a geração de relatórios e gráficos que indicam o perfil de demanda da infraestrutura do HUSFP, contribuindo para uma melhor administração dos recursos.

Na prototipação realizada nesta dissertação foi feito o monitoramento de uma unidade de óxido nitroso, utilizando um servidor de borda do EXEHDA, porém a arquitetura empregada foi planejada para permitir a expansão da quantidade de Servidores de Borda e sensores, possibilitando a coleta de dados de outros contextos de interesse do HUSFP.

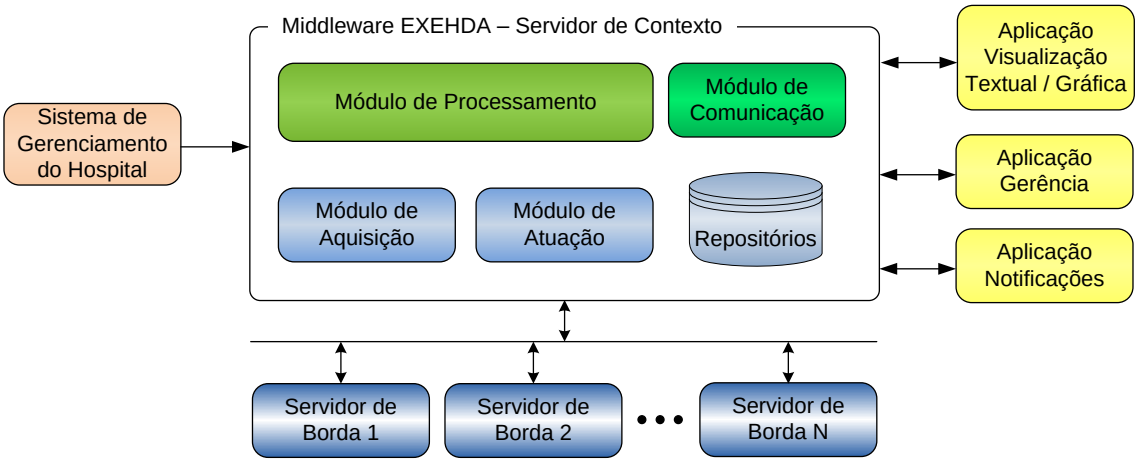

Figura 5.12 – Principais elementos envolvidos no monitoramento da infraestrutura hospitalar.

A parametrização do Servidor de Contexto do EXEHDA-IS, de acordo com as características de processamento contextual e identificação de situações necessárias para a aplicação, foi feita utilizando a API do Gerenciador de Fluxos de Processamento, através da qual foram criados os Fluxos que serão apresentados na seção a seguir.

5.3.1 Fluxos de Processamento

Foram criados dois Fluxos para realizar as etapas de processamento contextuais da aplicação. Na Figura 5.13 pode ser visto o Fluxo utilizado para o tratamento das postagens de vazão, cálculo de volume, identificação de situação de consumo e projeção de reabastecimento. Na Figura 5.14 tem-se o fluxo para processamento das postagens de temperatura.

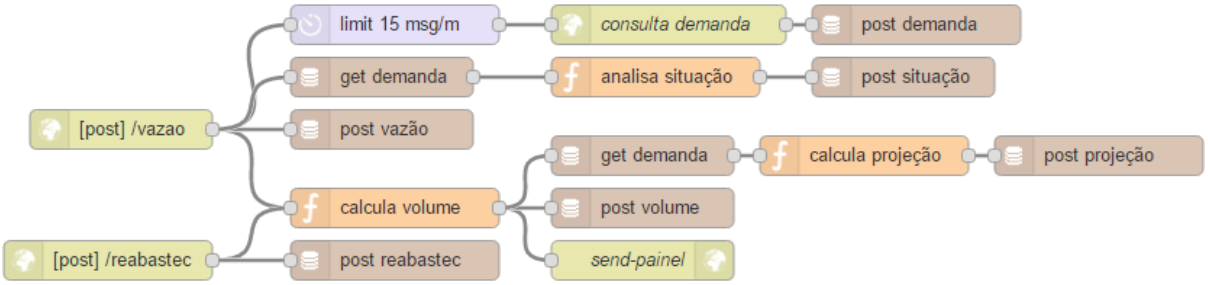

Figura 5.13 – Fluxo de Processamento Contextual para tratamento das postagens de vazão.

A recepção de uma mensagem com um novo valor de vazão dispara quatro ações de processamento, conforme Figura 5.13, executando:

- gravação do valor de vazão recebido no Repositório de Informações de Contexto;
- cálculo do volume estimado de óxido nitroso restante no cilindro e armazenamento do valor no Repositório de Contexto. A atualização do valor de volume dispara também: (I) comando de atualização do indicador de volume localizado juto ao Servidor de Borda; (II) cálculo da projeção de substituição do cilindro de óxido nitroso;
- análise da situação de consumo de óxido nitroso atual, identificando possíveis situações de desperdício baseado na demanda prevista para o momento;
- atualização da demanda prevista para o hospital em intervalos de, no mínimo 15 minutos, e armazenamento dos dados no Repositório de Contexto. Esta parte do fluxo realiza o aceso para uma API disponibilizada pelo sistema de gerenciamento do HUSFP. O vínculo deste fluxo com a postagem de valores de vazão permite o controle automático da consulta de demanda, visto que a mesma só precisa ser realizada caso a medição de vazão esteja ativa.

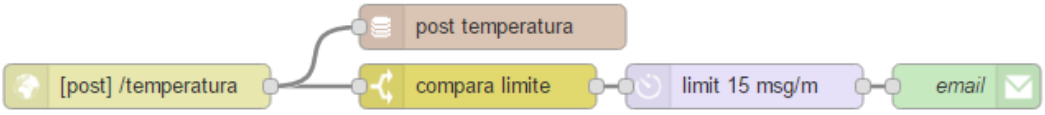

Figura 5.14 – Fluxo de Processamento Contextual para tratamento das postagens de temperatura.

A postagem de medições de temperatura, conforme Figura 5.14, dispara a ação de armazenamento do valor medido no Repositório de Informações de Contexto, e caso o valor medido esteja acima do limite máximo dispara um e-mail de notificação com intervalo de no mínimo 15 minutos enquanto a temperatura estiver acima do limite.

5.3.2 Aplicação de Visualização e Gerenciamento

Nesta seção é apresentada a aplicação desenvolvida para o gerenciamento e visualização dos dados gerados neste estudo de caso. A mesma é acessada através de *web browsers*.

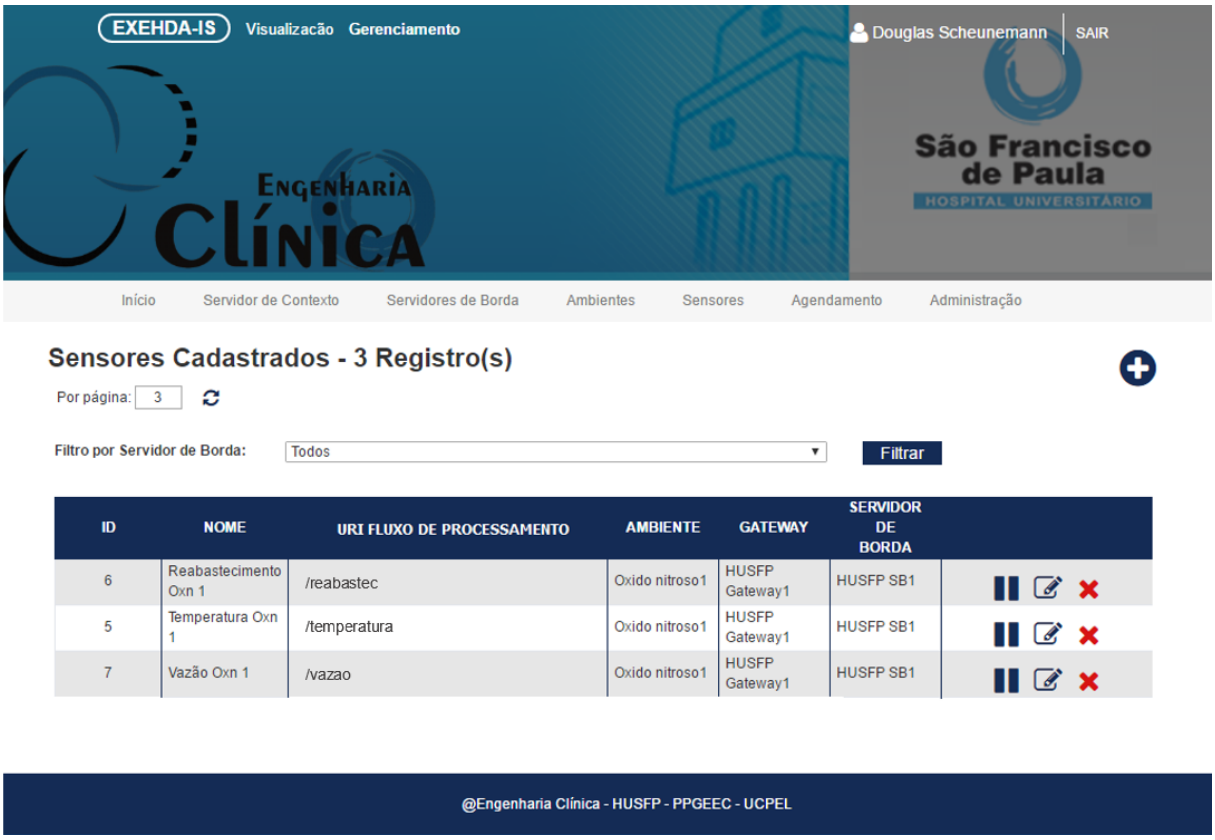

Figura 5.15 – Interface de gerenciamento de sensores.

A aplicação permite o cadastro de usuários com privilégios para administração e/ou visualização. No perfil de administrador é possível gerenciar o cadastro de Servidores de Borda, Contextos de Interesse e Sensores. Na 5.15 é apresentada a interface para gerenciamento de sensores, através da qual é vinculado do Fluxo de Processamento que será acionado quando um dado do respectivo sensor for recebido.

Através dos mecanismos para visualização de dados disponibilizados pela aplicação é possível escolher o contexto de interesse e o sensor para qual deseja-se analisar os valores. Na Figura 5.16 é mostrada esta interface, na qual está selecionado o Contexto “Óxido Nitroso” e Sensor “Vazão Oxn 1”. Após clicar no botão “Continuar” é exibido o gráfico mostrado na Figura 5.17.

Figura 5.16 – Interface para seleção de Contexto e Sensor.

Os dados apresentados no gráfico da Figura 5.17 representam o registro da medição de vazão para o período de duas semanas. Os dados foram gerados no servidor de borda considerando a demanda média do HUSFP. O uso do óxido nitroso apresenta picos que coincidem com a realização de procedimentos cirúrgicos seguidos de períodos sem consumo. A área do HUSFP suprida pelo cilindro monitorado possui sete pontos de utilização, podendo chegar a um consumo estimado de 105 l/min quando todos os pontos estiverem ativos.

Nota-se na Figura 5.17 a exibição do parâmetro "Situação". Este indica se o consumo atual de óxido nitroso corresponde à demanda fornecida pelo sistema de gerenciamento do HUSFP. Essa situação é classificada nos níveis: Baixo, Normal e Alto. Quando o consumo é classificado como "Alto", é indicada a ocorrência de possíveis situações de desperdício.

Figura 5.17 – Gráfico dos dados de vazão gerado através da interface de visualização.

O volume de óxido nitroso presente no cilindro monitorado é calculado pelo Fluxo de Processamento mostrado na 5.13. Na Figura 5.18 tem-se o gráfico gerado para o mesmo período da vazão mostrada na Figura 5.17. Os pontos onde o volume é aumentado indicam o reabastecimento do cilindro, informado através do painel de controle junto ao Servidor de Borda.

A provável data de reabastecimento do óxido nitroso é calculada a partir do volume atual do cilindro e da demanda futura informada pelo sistema de gerenciamento do HUSFP. Na Figura 5.18 pode ser vista a projeção calculada no item "Projeção Reabastecimento".

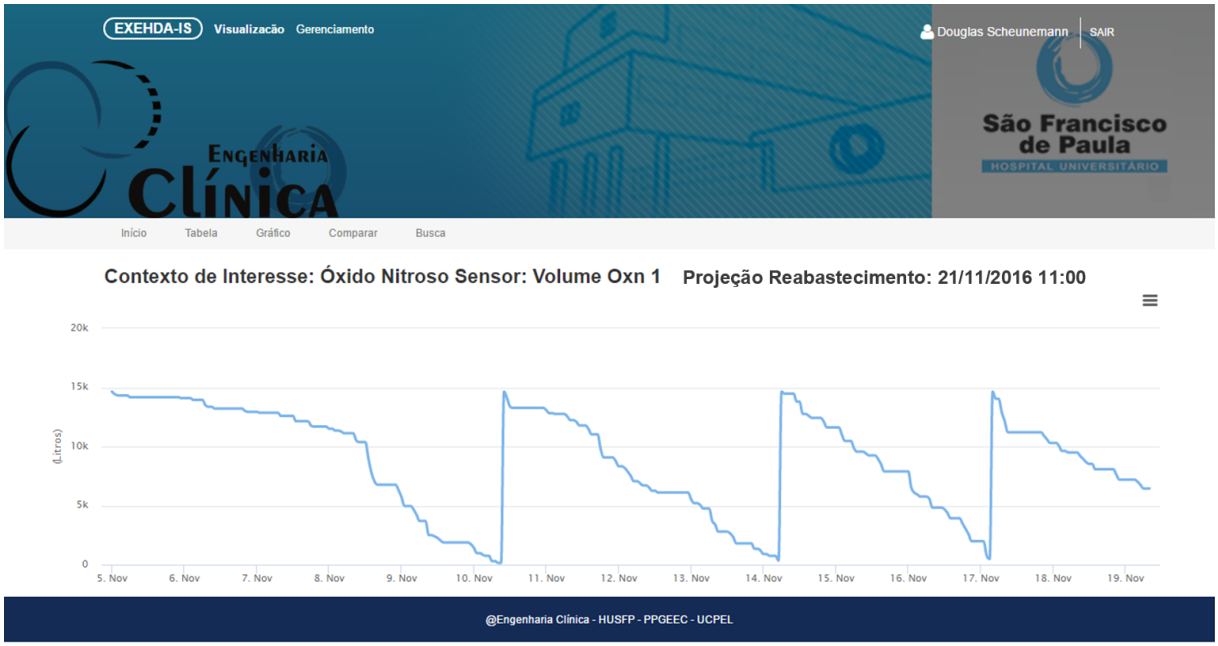

Figura 5.18 – Gráfico dos dados de volume gerado através da interface de visualização.

Para realizar a coleta de dados e prototipação do Servidor de Borda foram utilizados alguns elementos de hardware, descritos na seção a seguir.

5.3.3 Dispositivos de Hardware Empregados

Os Servidores de Borda previstos no sistema de monitoramento foram construídos utilizando o hardware **Raspberry Pi Modelo B+** (BROCK; BRUCE; CAMERON, 2013), mostrado na Figura 5.19. O mesmo é um computador desenvolvido pelo Laboratório de Computação da Universidade de Cambridge, que tem como principais características tamanho reduzido, baixo custo e baixo consumo energético, possuindo uma comunidade de usuários bastante ativa e que baseia-se nos princípios de software livre.

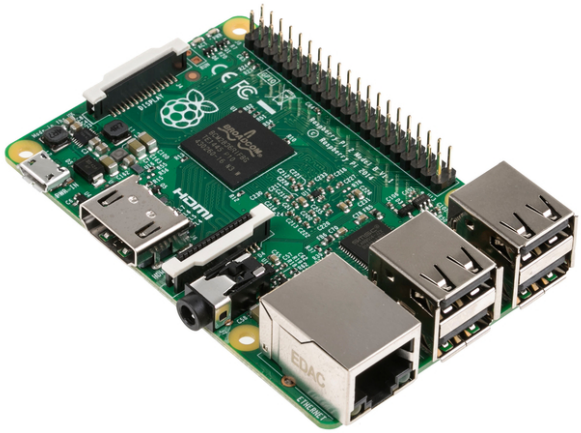

Figura 5.19 – Placa Raspiberry Pi Modelo B+.

A medição da vazão de óxido nitroso é feita utilizado o sensor de vazão **Honeywell HAFUHH0300L4AXT**, mostrado na Figura 5.20. Este sensor possibilita a medição de vazões de até 300 l/min em uma tubulação com pressão de 414 kPa, operando com precisão de 0,5 %FS até 43 l/min e 7,0 %FS em 300 l/min. A comunicação com o sensor é feita através do protocolo *Inter-Integrated Circuit* (I2C) (Honeywell, 2016).

Figura 5.20 – Sensor de vazão de gás Honeywell HAFUHH0300L4AXT.

A medição da temperatura do ambiente onde é instalado o cilindro de óxido nitroso foi feita utilizando o sensor DS18B20. O mesmo é acessado através do protocolo 1-Wire. Para realizar a leitura dos sensores de vazão e temperatura que utilizam interfaces I2C e 1-Wire, respectivamente, foi utilizado o hardware NodeMCU [3], conforme Figura 5.21. Este dispositivo é baseado no *System on Chip* (SoC) ESP8266, o qual possui um *core* de processamento ARM [4] de 80 MHz e hardware integrado para conexão em redes WiFi com suporte ao protocolo TCP/IP. Possui ainda, suporte para comunicação serial (SPI, I2C e UART), 16 pinos GPIO digitais e uma entrada analógica.

[3] https://github.com/nodemcu/nodemcu-firmware/wiki
[4] https://www.arm.com/

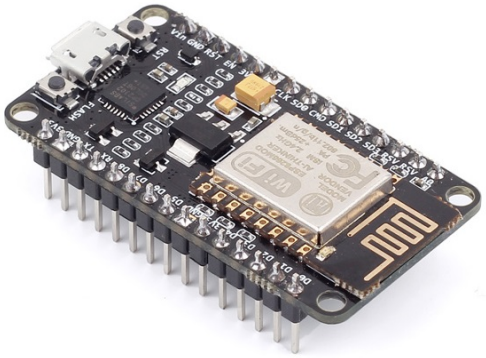

Figura 5.21 – Hardware NodeMCU.

5.3.4 Análise do Estudo de Caso

O estudo de caso em Ciência de Situação em Infraestrutura Hospitalar apresentado nesta seção, permitiu demostrar o uso de alguns recursos da arquitetura desenvolvida, além de demostrar como ferramentas de visualização e gerenciamento podem ser construídas sobre as suas APIs. Os Fluxos de Processamento criados para o processamento contextual e identificação de situações permitiram demostrar o uso do modelo *dataflow* de programação como estratégia de programação, em que funcionalidades da arquitetura são abstraídas através de Componentes de Processamento criados para o EXEHDA-IS.

5.4 Considerações Sobre o Capítulo

Neste capítulo foram apresentados os estudos de caso desenvolvidos para a avaliação da arquitetura do EXEHDA-IS, sendo destacadas as principais características, funcionalidades e objetivos das aplicações prototipadas. Também, foram descritas as tecnologias de software empregadas na prototipação das aplicações. Os cenários utilizados apresentam características típicas da IoT, detre elas: conexão com a internet, capacidade de sensoriamento e atuação, inteligência integrada e programabilidade. Estas características evidenciam o potencial de uso do EXEHDA-IS na IoT.

6 CONSIDERAÇÕES FINAIS

Neste capítulo são apresentas as principais conclusões relacionadas à concepção do EXEHDA-IS, bem como as publicações realizadas e as propostas para trabalhos futuros.

6.1 Principais Conclusões

No cenário em que estão previstos cerca de 40 bilhões de dispositivos conectados em 2020, a IoT vem ganhando notoriedade tanto em trabalhos acadêmicos quanto em âmbito comercial. Diversas áreas como protocolos, interfaces e *middlewares* vêm sendo estudadas e desenvolvidas.

Pesquisas na área apontam que o processamento contextual e a ciência de situação são ainda pouco exploradas em *middlewares* para IoT, dada a complexidade das etapas de processamento que precisam ser adicionadas para dar suporte a estas funcionalidades. Considerando isto, esta dissertação de mestrado teve como objetivo central o desenvolvimento de uma arquitetura para realizar o processamento contextual híbrido para a identificação de situações na IoT, a qual foi integrada com o *middleware* EXEHDA contribuindo para o seu Subsistema de Adaptação e Reconhecimento do Contexto.

Através da revisão de literatura foram evidenciadas as principais técnicas utilizadas para o processamento contextual, bem como os trabalhos relacionados com o tema desta dissertação. Esta análise proveu subsídios para a elaboração da proposta da arquitetura, partindo do estado da arte em ciência de situação para aplicações em IoT.

Foi proposto na arquitetura do EXEHDA-IS uma abordagem *dataflow* para prover um modelo híbrido de processamento contextual, visando potencializar as características de técnicas de aprendizado de máquina e especificação de regras. A utilização de uma abordagem *dataflow* para programação do processamento contextual permite a criação de Fluxos de Processamento Contextuais em um modelo com alto nível de abstração, facilitando a inferência de situações. Além disso, a programação *dataflow* empregada nesta dissertação permite tornar transparente ao programador questões de concorrência e sincronização inerentes a execução dos Componentes de Processamento.

Os estudos de caso desenvolvidos permitiram avaliar a arquitetura do EXEHDA-IS frente a aplicação de modelos de processamento contextuais híbridos. Possibilitaram ainda, demostrar a utilização do Gerenciador de Fluxos de Processamento, acessado através de uma API REST. De maneira geral, os estudos de caso mostram a relevância da arquitetura proposta

frente a cenários de aplicação típicos da IoT, nos quais as fontes de dados são distribuídas e heterogêneas e as aplicações devem ser carregadas de forma mínima com tarefas de processamento contextual e identificação de situações.

6.2 Publicações Realizadas

- **SCHEUNEMANN, D.**; LOPES, J. L. B.; REISER, R.; YAMIN, A. C.; GEYER, C. Middleware Architecture for Supporting a Hybrid Processing of Context Data Targeted to Detection of Situations in Ubicomp. In: **12th International FLINS Conference - Conference on Uncertainty Modelling in Knowledge Engineering and Decision Making**. Roubaix, France: FLINS, 2016.

- **SCHEUNEMANN, D.**; PARREIRA, W. D.; YAMIN, A. C.; LOPES, J. L. B.; GEYER, C.. Cardiac Rehabilitation: an IoT Architecture Exploring Situation Awareness Based on Open-Source Technologies. In: **XXXIV Simpósio Brasileiro de Telecomunicações e Processamento de Sinais**. Santarém - PA: SBRT, 2016.

- **SCHEUNEMANN, D.**; LOPES, J. L. B.; YAMIN, A. C.; GEYER, C. Uma Contribuição à Reabilitação Cardíaca Explorando a Identificação de Situações na IoT. In: **SEMISH - 43º Seminário Integrado de Software e Hardware**. Porto Alegre - RS: SEMISH, 2016. p. 1772–1782.

- **SCHEUNEMANN, D.**; LOPES, J. L. B.; YAMIN, A. C.; GEYER, C. Identificação de Situações de Risco para Pacientes em Reabilitação Cardíaca Explorando uma Arquitetura de Software na Internet das Coisas. In: **WIM - XVI Workshop de Informática Médica**. Porto Alegre - RS: WIN, 2016. p. 1772–1782.

- **SCHEUNEMANN, D.**; LOPES, J. L. B.; ; REISER, R.; YAMIN, A. C.; GEYER, C. Identificação de Situações em eHealth Explorando uma Abordagem Híbrida Baseada em Lógica Fuzzy e Árvore de Decisão. In: **IV CBSF - Congresso Brasileiro de Sistemas Fuzzy**. Campinas - SP: CBSF, 2016.

- **SCHEUNEMANN, D.**; LOPES, J. L. B.; YAMIN, A. C. Explorando o Processamento Híbrido de Contexto no Reconhecimento de Atividades. In: **XVI Escola Regional de Alto Desempenho do Estado do Rio Grande do Sul**. São Leopoldo - RS: ERAD/RS, 2016.

- **SCHEUNEMANN, D.**; YAMIN, A. C.Uma Arquitetura para Processamento Híbrido de Contexto Direcionada às Aplicações Cientes de Situação na IoT. In: **Salão Universitário UCPEL**. Pelotas - RS: UCPEL/RS, 2016.

- **SCHEUNEMANN, D.**; YAMIN, A. C. Sistemas Ubíquos para Saúde: Uma Revisão de Técnicas para Consciência de Situação. In: **Salão Universitário UCPEL**. Pelotas - RS: UCPEL/RS, 2015.

Um registro dos diferentes trabalhos desenvolvidos ao longo do Mestrado está disponível em <http://olaria.ucpel.tche.br/douglas/>.

6.3 Trabalhos Futuros

A arquitetura proposta nesta dissertação de mestrado se mostrou promissora frente a cenários de aplicações da IoT e também como contribuição para a arquitetura do Servidor de Contexto do *middleware* EXEHDA. Durante o seu desenvolvimento foram evidenciadas algumas possibilidades continuidade do trabalho, visando a expansão de funcionalidades da arquitetura e também dos estudos de caso, incluindo:

- **interface gráfica para a API do Gerenciador de Fluxos de Processamento** – o acesso a API REST criada para o gerenciador em questão foi feita através de ferramentas para comunicação por protocolo HTTP. O desenvolvimento de uma interface gráfica para realizar o gerenciamento dos fluxos de processamento facilitará a utilização da API em aplicações desenvolvidas utilizando o *middleware* EXEHDA;

- **interface para criação de regras fuzzy** – o uso da lógica fuzzy como ferramenta para criação de modelos de raciocínio baseados em especificação se mostrou promissor no estudo de caso apresentado na Seção 5.2. Desta forma a criação de uma ferramenta gráfica para a criação das funções de pertinência e regras para inferência pode potencializar o seu uso em novas aplicações;

- **expansão do mecanismo para identificação de atividades** – a atividade realizada por usuários de aplicações em IoT é uma importante informação de contexto, que permite criar aplicações mais proativas que forneçam serviços customizados e oportunos. No estudo de caso apresentado na seção 5.2 foi apresentado um mecanismo baseado em aprendizado para identificar as atividades “Repouso”, “Movimento Lento”

e “Movimento Rápido”. Expandir a quantidade de atividades identificadas, avaliar outras técnicas de processamento, combinar dados de outros tipos de sensores e realizar experimentos com voluntários possibilitaria um refinamento maior das atividades identificadas proporcionado a criação de novos cenários de uso.

REFERÊNCIAS

ALLEN, J. F.; FERGUSON, G. Actions and events in interval temporal logic. **Journal of Logic and Computation**, v. 4, n. 5, p. 531–579, 1994. ISSN 0955792X.

AUGUSTO, J. C. et al. Management of Uncertainty and Spatio-Temporal Aspects for Monitoring and Diagnosis in a Smart Home. **International Journal of Computational Intelligence Systems**, v. 1, n. 4, p. 361, 2008. ISSN 1875-6883.

AZZARA, A. et al. PyoT, a macroprogramming framework for the Internet of Things. **Proceedings of the 9th IEEE International Symposium on Industrial Embedded Systems, SIES 2014**, n. i, p. 96–103, 2014.

BELLAVISTA, P. et al. A survey of context data distribution for mobile ubiquitous systems. **ACM Computing Surveys**, v. 44, n. 4, p. 1–45, 2012. ISSN 03600300.

BETTINI, C. et al. A survey of context modelling and reasoning techniques. **Pervasive and Mobile Computing**, Elsevier B.V., v. 6, n. 2, p. 161–180, 2010. ISSN 15741192.

BIBRI, S. E. **The Human Face of Ambient Intelligence**. [S.l.: s.n.], 2015. 197–257 p. ISBN 978-94-6239-129-1.

BLACKSTOCK, M.; LEA, R. Towards a distributed data flow platform for the web of things. **Proceedings of the 5th International Workshop on Web of Things**, p. 34–39, 2014.

BORGES, L. E. **Python para Desenvolvedores**. 2. ed. Rio de Janeiro: [s.n.], 2010. 360 p. ISBN 9788590945116.

BROCK, J. D.; BRUCE, R. F.; CAMERON, M. E. Changing the world with a Raspberry Pi. **Journal of Computing Sciences in Colleges**, Consortium for Computing Sciences in Colleges, v. 29, n. 2, p. 151–153, 2013. ISSN 1937-4771.

CHIBELUSHI, C. An Overview and Concerns on IoT in Healthcare Applications ; The Role of AI and Pattern Recognition. **IJICTT**, v. 1, n. 1, p. 39–52, 2014.

CHIHANI, B.; BERTIN, E.; CRESPI, N. Programmable context awareness framework. **Journal of Systems and Software**, Elsevier Inc., v. 92, n. 1, p. 59–70, 2014. ISSN 01641212.

COSTA, P. D. et al. Situations in conceptual modeling of context. **Proceedings - 2006 10th IEEE International Enterprise Distributed Object Computing Conference Workshops, EDOCW2006**, n. i, 2006.

ECMA-404. **The JSON Data Interchange Format**. 2013. 8 p. Available from Internet: <http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf>.

ENDSLEY, M. R. Measurement of Situation Awareness in Dynamic Systems. **Human Factors: The Journal of the Human Factors and Ergonomics Society**, v. 37, n. 1, p. 65–84, 1995. ISSN 00187208.

FIELDING, R. T. **Architectural Styles and the Design of Network-based Software Architectures**. 162 p. Thesis (PhD) — University of California, 2000.

FORKAN, A.; KHALIL, I.; TARI, Z. CoCaMAAL: A cloud-oriented context-aware middleware in ambient assisted living. **Future Generation Computer Systems**, Elsevier B.V., v. 35, p. 114–127, 2014. ISSN 0167739X.

FOSLER-LUSSIER, E. Markov Models and Hidden Markov Models: A Brief Tutorial. **Ca Tr-98-041**, v. 1198, n. 510, p. 132–141, 1998.

GIANG, N. K. et al. Distributed Data Flow: A Programming Model for the Crowdsourced Internet of Things. **Proceedings of the Doctoral Symposium of the 16th International Middleware Conference**, p. 4:1–4:4, 2015.

GUINARD, D.; TRIFA, V.; WILDE, E. A resource oriented architecture for the Web of Things. **Proc. of 2010 Internet of Things (IOT), 2010**, p. 1–8, 2010.

HAGHIGHI, D. P. et al. Situation-Aware Mobile Health Monitoring. **ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering)**, p. 248–256, 2014.

HEO, S. et al. IoT-MAP : IoT Mashup Application Platform for the Flexible IoT Ecosystem. **Internet of Things (IOT), 2015 5th International Conference on the**, p. 163–170, 2015.

Honeywell. **Honeywell ZephyrTM Digital Airflow Sensors: HAF Series–High Accuracy10 SLPM, 15 SLPM, 20 SLPM, 50 SLPM, 100 SLPM,200 SLPM or 300 SLPM Datasheet**. 2016. 16 p. Available from Internet: <http://sensing.honeywell.com/index.php?ci{_}id=144>.

KHODADADI, F.; DASTJERDI, A. V.; BUYYA, R. Simurgh: A framework for effective discovery, programming, and integration of services exposed in IoT. **Recent Advances in Internet of Things (RIoT), 2015 International Conference on**, n. April, p. 1–6, 2015.

KNAPPMEYER, M. et al. Survey of context provisioning middleware. **IEEE Communications Surveys and Tutorials**, v. 15, n. 3, p. 1492–1519, 2013. ISSN 1553877X.

KORPIPÄÄ, P. et al. Bayesian approach to sensor-based context awareness. **Personal and Ubiquitous Computing**, v. 7, n. 2, p. 113–124, 2003. ISSN 16174909.

KWAPISZ, J. R.; WEISS, G. M.; MOORE, S. a. Activity recognition using cell phone accelerometers. **ACM SIGKDD Explorations Newsletter**, v. 12, p. 74, 2011. ISSN 19310145.

LARA, O. D.; LABRADOR, M. a. A Survey on Human Activity Recognition using Wearable Sensors. **IEEE Communications Surveys & Tutorials**, p. 1–18, 2012. ISSN 1553-877X.

LIM, C. H.; VATS, E.; CHAN, C. S. Fuzzy human motion analysis: A review. **Pattern Recognition**, v. 48, n. 5, p. 1773–1796, 2014. ISSN 00313203.

LOKE, S. W. Logic Programming for Context-Aware Pervasive Computing. **Proceedings of the IEEE/WIC/ACM International Conference of Web Intelligence**, n. 2, p. 1–7, 2004.

LOPES, J. et al. A middleware architecture for context-aware adaptation in. **Journal of Universal Computer Science**, v. 20, n. 9, p. 1327–1351, 2014. ISSN 09486968.

LOPES, J. et al. A Distributed Architecture for Supporting Context-Aware Applications in UbiComp. **2014 IEEE 28th International Conference on Advanced Information Networking and Applications**, p. 584–590, 2014.

MINERVA, R.; BIRU, A.; ROTONDI, D. Towards a definition of the Internet of Things (IoT). **IEEE Internet of Things**, n. 1, p. 1–86, 2015.

MONGODB, I. **The MongoDB 3.2 Manual — MongoDB Manual 3.2**. 2016. Available from Internet: <https://docs.mongodb.com/manual/>.

NEGRÃO, C. E.; BARRETO, A. C. P. **Cardiologia do Exercício: do Atleta ao Cardiopata**. 3. ed. Barueri, SP - Brazil: Manole, 2010. 725 p. ISBN 978-85-204-3075-0.

PERERA, C. et al. Context Aware Computing for The Internet of Things : A Survey. **IEEE COMMUNICATIONS SURVEYS & TUTORIALS**, X, n. X, p. 1–41, 2013.

PÉREZ, J. L. et al. The COMPOSE API for the Internet of Things. **World Wide Web**, p. 971–976, 2014.

RABELO, D.; GIL, C.; ARAÚJO, S. D. Reabilitação cardíaca com ênfase no exercício: uma revisão sistemática. **Revista Brasileira de Medicina do Esporte**, v. 12, n. 5, p. 279–285, 2006.

RABINER, L. R. A tutorial on hidden Markov models and selected applications in speech recognition. **Proceedings of the IEEE**, v. 77, n. 2, p. 257–286, 1989. ISSN 00189219.

RAVEENDRANATHAN, N. et al. From modeling to implementation of virtual sensors in body sensor networks. **IEEE Sensors Journal**, v. 12, n. 3, p. 583–593, 2012. ISSN 1530437X.

RAZZAQUE, M. A. et al. Middleware for Internet of Things: a Survey. **Internet of Things Journal, IEEE**, v. 3, n. 1, p. 70–95, 2016. ISSN 2327-4662.

SHARMA, M. Fuzzy Logic Based Handover. **International Journal of Ad hoc, Sensor & Ubiquitous Computing (IJASUC)**, v. 3, n. 4, p. 21–29, 2012.

SHAW, I.; SIMõES, M. **Controle e Modelagem Fuzzy**. 2. ed. [S.l.: s.n.], 2007. 200 p. ISBN 978-85-212-0416-9.

SOBREVILLA, P.; MONTSENY, E. Fuzzy Sets in Computer Vision : an Overview. **Mathw. Soft Computing**, v. 10, p. 71–83, 2003.

SOUSA, T. B. Dataflow programming concept, languages and applications. In: **Doctoral Symposium on Informatics Engineering**. [S.l.: s.n.], 2012.

VERMESAN, O. et al. Internet of Things Strategic Research Roadmap. 2009.

WITTEN, I. H.; FRANK, E.; HALL, M. a. **Data Mining: Practical Machine Learning Tools and Techniques, Third Edition**. 3. ed. [S.l.: s.n.], 2011. 664 p. ISSN 14337851. ISBN 9780123748560.

YAMIN, A. C. **Arquitetura para um Ambiente de Grade Computacional Direcionado às Aplicações Distribuídas , Móveis e Conscientes do Contexto da Computação Pervasiva**. 195 p. Thesis (PhD), 2004.

YANG, J.-Y.; WANG, J.-S.; CHEN, Y.-P. Using acceleration measurements for activity recognition: An effective learning algorithm for constructing neural classifiers. **Pattern Recognition Letters**, v. 29, n. 16, p. 2213–2220, 12 2008. ISSN 01678655.

YE, J.; DOBSON, S.; MCKEEVER, S. Situation identification techniques in pervasive computing: A review. **Pervasive and Mobile Computing**, Elsevier B.V., v. 8, n. 1, p. 36–66, 2012. ISSN 15741192.

YUAN, B.; HERBERT, J. Context-aware hybrid reasoning framework for pervasive healthcare. **Personal and Ubiquitous Computing**, v. 18, n. 4, p. 865–881, 2014. ISSN 1617-4909.