Uma Abordagem Dataflow para Processamento Contextual na Internet das Coisas *

Douglas Scheunemann[1], Adenauer Yamin[1], Renata Reiser[2], João Lopes[3], Cláudio Geyer[3]

[1]Universidade Católica de Pelotas (UCPEL) – Pelotas – RS – Brasil
[2]Universidade Federal de Pelotas (UFPEL) – Pelotas – RS – Brasil
[3]Universidade Federal do Rio Grande do Sul (UFRGS) – Porto Alegre – RS – Brasil

scheunemann.d.a@gmail.com, adenauer.yamin@ucpel.edu.br,
reiser@inf.ufpel.edu.br,
{jlblopes, geyer}@inf.ufrgs.br

Abstract. *The objective of this work is the design of an architecture focused on the composition of contextual processing flows to provide situation awareness for IoT applications. This work considers the use of a dataflow programming approach integrated with the EXEHDA middleware. A case study in the health area was carried out to evaluate the architecture.*

Resumo. *O objetivo deste trabalho é a concepção de uma arquitetura voltada para a composição de fluxos de processamento contextuais, provendo ciência de situação para aplicações na IoT. Este trabalho explora o uso de uma abordagem de programação dataflow integrada ao middleware EXEHDA. Como forma de avaliação da arquitetura, foi executado um estudo de caso na área da saúde.*

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 originalmente 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. Estes dispositivos podem fornecer informações de contexto, as quais através de técnicas de processamento contextual, permitem a identificação de situações. Nesse sentido, a ciência de situação representa a capacidade 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].

A principal contribuição deste trabalho é a concepção de uma arquitetura de software, integrada ao *middleware* EXEHDA (*Execution Environment for Highly Distributed Applications*) [Lopes et al. 2014], nomeada EXEHDA-IS (*EXEHDA – IoT Situations*)

*O presente trabalho foi realizado com apoio do Programa Nacional de Cooperação Acadêmica da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – CAPES/Brasil.

voltada para a composição de mecanismos do processamento contextual com o intuito de prover ciência de situação para aplicações no cenário da IoT.

O principal diferencial da arquitetura proposta, quando comparada com a de outros middlewares para IoT, é uso de uma abordagem *dataflow* para programação do processamento contextual, permitindo a composição de fluxos de processamento, os quais englobam técnicas de raciocínio baseadas em especificação e em aprendizagem e abstraem a distribuição física dos dispositivos.

A arquitetura proposta foi avaliada através de um estudo na área de saúde com foco em reabilitação cardíaca. Este estudo de caso permitiu a exploração de funcionalidades relativas à parametrização do processamento contextual utilizando a abordagem de programação *dataflow* e também a aplicação de um modelo raciocínio para identificação de situações utilizando lógica fuzzy e árvores de decisão.

2. EXEHDA-IS: Visão Geral e Funcionalidades

O objetivo do EXEHDA-IS é prover uma arquitetura de software que permita a **composição** e o **gerenciamento** de mecanismos para processamento contextual executado no servidor de contexto do *middleware* EXEHDA, combinando técnicas de aprendizado e especificação de regras. Na Seção 2.1 será introduzida a arquitetura do Servidor de Contexto do EXEHDA e nas Seções 2.2 e 2.3 serão apresentadas a arquitetura do módulo de processamento e a abordagem de programação *dataflow* propostas neste trabalho.

2.1. Arquitetura do Servidor de Contexto

O *middleware* EXEHDA possui dois tipos de servidores: (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. Estes servidores são alocados em células no ambiente gerenciado pelo EXEHDA, onde cada célula possui um Servidor de Contexto e pode possuir vários Servidores de Borda.

O Servidor de Contexto é formado por cinco módulos: Aquisição, Atuação, Notificação, Comunicação e Processamento [Lopes et al. 2014], os quais podem ser visualizados na Figura 1. Os elementos destacados na figura representam as partes onde estão as principais contribuições deste trabalho, 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.

O **Módulo de Aquisição** é responsável pela captura das informações contextuais, coletadas pelos Servidores de Borda considerando sensores lógicos e físicos. Através do **Módulo de Atuação** é controlada a ativação, desativação e configuração dos atuadores, como consequência de uma notificação de outros módulos do Servidor de Contexto. O **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. Por sua vez, o **Módulo de Comunicação** é utilizado por Servidores de Contexto remotos e/ou aplicações para solicitação de situações constituídas por contextos de interesse correlacionados, dados contextuais e/ou o disparo de atuadores.

O **Módulo de Processamento** tem como principal função realizar as tarefas pertinentes ao tratamento das informações contextuais, bem como dos eventos para identificar

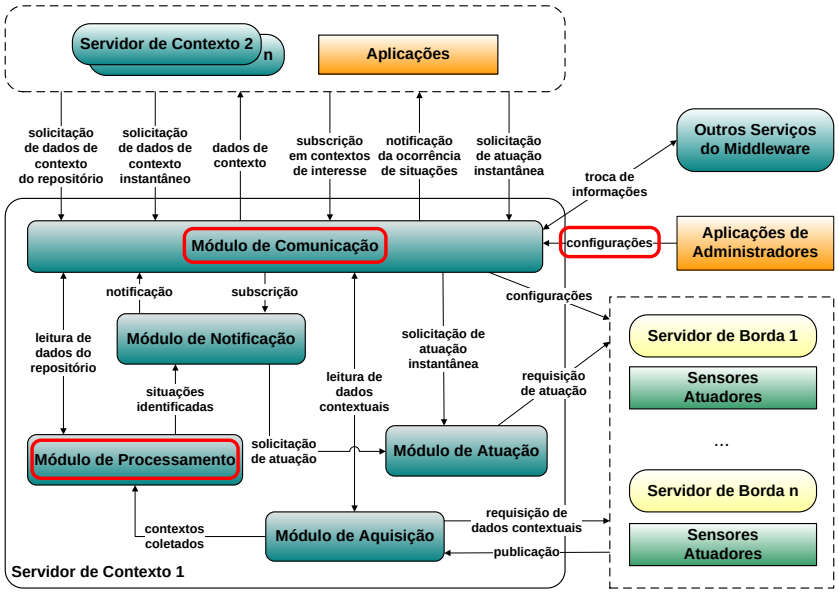

Figura 1. Arquitetura do servidor de contexto.

situações de interesse das aplicações. Para execução destas tarefas, é proposto neste trabalho o uso de um modelo de processamento contextual que permite a combinação de técnicas baseadas em aprendizado e especificação de regras, apresentado na Seção 2.2.

A **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 *dataflow* proposta para esta interface é apresentada na Seção 2.3.

2.2. Arquitetura do Módulo de Processamento

A arquitetura do Módulo de Processamento proposto para o EXEHDA-IS foi elaborada visando o processamento de contexto através da combinação de técnicas baseadas em aprendizado e especificação de regras. Desta forma, possui componentes que atendem a demandas relacionadas com estes dois tipos de técnicas e também com o 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 permite que um número maior de situações possam ser identificadas, beneficiando-se do aprendizado do sistema e também da experiência de especialistas [Ye et al. 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, descritos a seguir.

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. 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 2.3.

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

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

Por sua vez, a etapa 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. Através de um processo de Normalização os valores numéricos destes dados contextuais são transformados para o domínio da técnica de especificação empregada.

A arquitetura do Módulo de Processamento do EXEHDA-IS é manipulada através de APIs e componentes de software que permitem compor os Fluxos de Processamento Contextuais, conforme descrito na Seção 2.3.

2.3. Abordagem *Dataflow* para Programação do Processamento Contextual

O método para programação do processamento contextual proposto para o EXEHDA-IS considera o uso de um modelo *dataflow* [Sousa 2012]. Tendo por base este modelo de programação, é buscado o 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. 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.

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 3, o encapsulamento dos dados das mensagens foi feito utilizando a linguagem de marcação JSON.

Na Figura 2 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 cadastrado 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 URI definida na sua criação, que será utilizada para endereçar as mensagens para o fluxo de processamento.

Como pode ser visto na Figura 2, as terminações dos Fluxos de Processamento acionam APIs do EXEHDA-IS, representadas também por Componentes de Processamento. Na Figura 2 são referenciadas as APIs do Repositório de Informações de Contexto e do Gerenciados de Fluxos de Processamento, as 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.

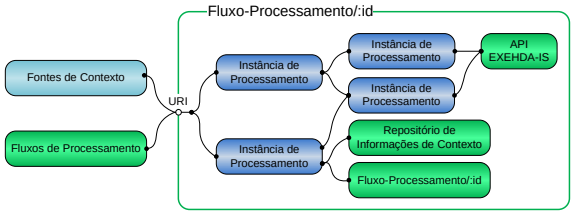

Figura 2. Modelo de processamento para a composição dos Fluxos de Processamento Contextual.

3. EXEHDA-IS: Avaliação da Arquitetura

Para realizar a avaliação das funcionalidades da arquitetura foi prototipada uma aplicação para monitoramento de pacientes em reabilitação cardíaca após um acidente vascular. 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 et al. 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 and 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.

A aplicação desenvolvida 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 3 pode ser visto o fluxo de coleta e processamento de dados contextuais proposto.

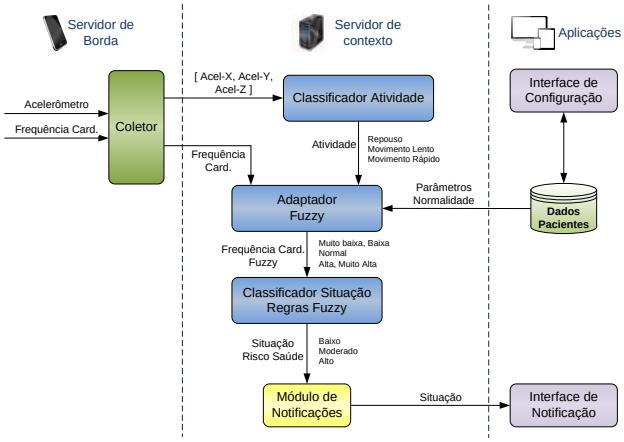

Figura 3. Visão geral do estudo de caso.

A identificação da atividade física do paciente foi feita utilizando algoritmos de classificação baseados em técnicas de aprendizado executadas sobre os sinais coletados a partir acelerômetro 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.

3.1. 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 et al. 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.

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 and Labrador 2012].

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. Foi utilizado um conjunto com 15 características do sinal no processo de classificação, que são 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 classificador baseado em Árvore de Decisão obteve a melhor precisão dentre os quatro classificadores testados, com 90,61 % de acerto. Os classificadores utilizando Rede Bayesiana, Rede Neural e Regressão Lógica, obtiveram 88,68 %, 89,01 % e 85,65 % como taxa de acerto, respectivamente.

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; e **Movimento Rápido**: correndo.

Utilizando a classificação agrupada, 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 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 4.

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 sobre a ferramenta Node-RED para este estudo de caso.

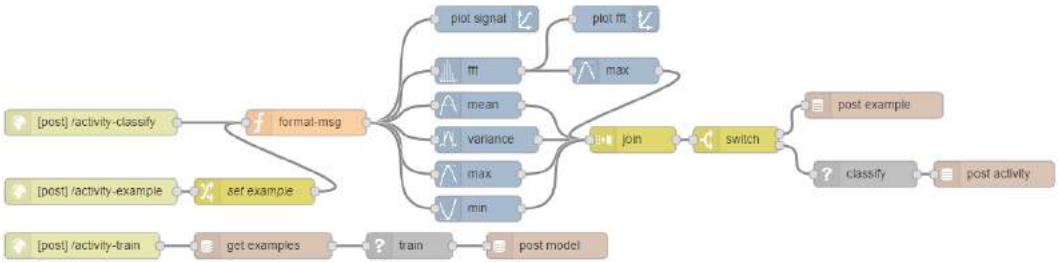

Figura 4. Fluxo de processamento voltado para a identificação de atividades.

3.2. 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 3.1.

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 3, 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 and Montseny 2003].

A inferência foi feita aplicado o método Mamdani, no qual são previstos quatro módulos: *fuzzificação*, base de regras, inferência e *defuzzificação* [Shaw and 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 and Simões 2007].

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*.

4. Trabalhos Relacionados

O estudo dos trabalhos relacionados foi feito considerando as seguintes características do EXEHDA-IS: (a) suporte para a criação e gerenciamento de fluxos de processamento contextual; (b) processamento de contexto com suporte a diferentes modelos de aprendizado na identificação de situações; (c) suporte à interação com sistemas e serviços da IoT; e (d) possibilidade de uso em diferentes domínios de aplicação.

O projeto CARA - *Context-Aware Real-time Assistant* [Yuan and 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 para processamento contextual, no qual técnicas para especificação de regras são combinadas com um algoritmo de aprendizado baseado em casos.

A arquitetura chamada Simurgh [Khodadadi et al. 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.

No *framework* proposto por [Chihani et al. 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* em que é suposto um compartilhamento do modelo contextual, o que em muitos casos limita a sua utilização em diferentes domínios de aplicação.

O *middleware* CoCaMALL – *A cloud-oriented context-aware middleware in ambient assisted living* [Forkan et al. 2014] é baseado em tecnologia de computação em nuvem com arquitetura orientada por serviços – 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 Tabela 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 1. Comparação dos trabalhos relacionados.

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

Conforme os dados apresentados na Tabela 1, nota-se que nenhuma das plataformas analisadas atende a todos os requisitos considerados, sendo que a abordagem para processamento contextual baseada em diferentes técnicas de aprendizado foi atendida plenamente somente pelo *middleware* CoCaMAAL.

As plataformas independentes de domínio de aplicação não oferecem suporte para o processamento de contexto baseado em aprendizado. Desta forma, uma lacuna de pesquisa nos trabalhos estudados é a ausência de uma arquitetura que combine modelos de processamento de contexto baseados em aprendizado com mecanismos para suporte à programação de fluxos de processamento contextuais, como é proposto no EXEHDA-IS.

5. Considerações Finais

A principal contribuição deste trabalho é a concepção da arquitetura EXEHDA-IS, a qual provê uma abordagem *dataflow* para programação do processamento contextual, visando à identificação de situações. Esta abordagem explora a combinação de técnicas baseadas em aprendizado de máquina e especificação de regras, com o intuito de potencializar suas cateterísticas para identificação de situações. Nesse sentido, a utilização da abordagem *dataflow* permite a criação de Fluxos de Processamento Contextuais em um modelo com alto nível de abstração, facilitando a inferência de situações.

O estudo de caso desenvolvido permitiu avaliar a arquitetura do EXEHDA-IS frente a aplicação de modelos de processamento contextuais baseados em aprendizado e em especificação de regras. Possibilitou ainda, demostrar a utilização do Gerenciador de Fluxos de Processamento. De maneira geral, o estudo de caso mostrou 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.

Referências Bibliográficas

Bellavista, P., Corradi, A., Fanelli, M., and Foschini, L. (2012). A survey of context data distribution for mobile ubiquitous systems. *ACM Computing Surveys*, 44(4):1–45.

Bibri, S. E. (2015). *The Human Face of Ambient Intelligence*, volume 9.

Chihani, B., Bertin, E., and Crespi, N. (2014). Programmable context awareness framework. *Journal of Systems and Software*, 92(1):59–70.

Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California.

Forkan, A., Khalil, I., and Tari, Z. (2014). CoCaMAAL: A cloud-oriented context-aware middleware in ambient assisted living. *Future Generation Computer Systems*, 35:114–127.

Khodadadi, F., Dastjerdi, A. V., and Buyya, R. (2015). 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*, (April):1–6.

Kwapisz, J. R., Weiss, G. M., and Moore, S. a. (2011). Activity recognition using cell phone accelerometers. *ACM SIGKDD Explorations Newsletter*, 12:74.

Lara, O. D. and Labrador, M. a. (2012). A Survey on Human Activity Recognition using Wearable Sensors. *IEEE Communications Surveys & Tutorials*, pages 1–18.

Lopes, J., Souza, R., Geyer, C., Costa, C., Barbosa, J., and Augustin, I. (2014). A middleware architecture for context-aware adaptation in. *Journal of Universal Computer Science*, 20(9):1327–1351.

Negrão, C. E. and Barreto, A. C. P. (2010). *Cardiologia do Exercício: do Atleta ao Cardiopata*. Manole, Barueri, SP - Brazil, 3 edition.

Perera, C., Zaslavsky, A., Christen, P., and Georgakopoulos, D. (2013). Context Aware Computing for The Internet of Things : A Survey. *IEEE COMMUNICATIONS SURVEYS & TUTORIALS*, X(X):1–41.

Rabelo, D., Gil, C., and Araújo, S. D. (2006). Reabilitação cardíaca com ênfase no exercício: uma revisão sistemática. *Revista Brasileira de Medicina do Esporte*, 12(5):279–285.

Shaw, I. and Simões, M. (2007). *Controle e Modelagem Fuzzy*. 2 edition.

Sobrevilla, P. and Montseny, E. (2003). Fuzzy Sets in Computer Vision : an Overview. *Mathw. Soft Computing*, 10:71–83.

Sousa, T. B. (2012). Dataflow programming concept, languages and applications. In *Doctoral Symposium on Informatics Engineering*, number January.

Ye, J., Dobson, S., and McKeever, S. (2012). Situation identification techniques in pervasive computing: A review. *Pervasive and Mobile Computing*, 8(1):36–66.

Yuan, B. and Herbert, J. (2014). Context-aware hybrid reasoning framework for pervasive healthcare. *Personal and Ubiquitous Computing*, 18(4):865–881.