Uma Arquitetura Distribuída Direcionada à Consciência de Contexto na Computação Ubíqua

João L. B. Lopes[2], Márcia Z. Gusmão[1], Patricia T. Davet[1], Rodrigo S. Souza[2], Ana M. Pernas[1], Adenauer C. Yamin[1], Cláudio F. R. Geyer[2]

[1]Universidade Federal de Pelotas (UFPel)

[2]Universidade Federal do Rio Grande do Sul (UFRGS)

{jlblopes, rssouza, geyer}@inf.ufrgs.br
{mzgusmao, pdavet, marilza, adenauer}@inf.ufpel.edu.br

Abstract. *One of the major research challenges of Ubiquitous Computing (UbiComp) is related to the need of the applications being aware of their context of interest, and when appropriate, respond to it. This paper presents an architecture for context awareness, called EXEHDA-UC (Ubiquitous Context-awareness). We consider that the main contribution of this work is an architecture that supports the managing of the acquisition, storage, and processing of context data, in a distributed way, independently of the application, in an autonomic perspective and rule-based. To assess the functionality of the EXEHDA-UC, we present a case study, highlighting the prototypes and tests performed.*

Resumo. *Um dos principais desafios de pesquisa da Computação Ubíqua (Ubicomp) está relacionado à necessidade das aplicações terem consciência do seu contexto de interesse, e quando for o caso, reagir ao mesmo. Este artigo apresenta uma arquitetura para consciência do contexto, denominada EXEHDA-UC (Ubiquitous Context awareness). Considera-se como principal contribuição deste trabalho a concepção de uma arquitetura para suporte ao gerenciamento da aquisição, armazenamento e processamento dos dados de contexto, de forma distribuída, independente das aplicações, em uma perspectiva autonômica baseada em regras. Para avaliar as funcionalidades do EXEHDA-UC é apresentado um estudo de caso, destacando os protótipos e testes realizados.*

1. Introdução

Na Computação Ubíqua as aplicações precisam ter consciência do seu contexto de interesse. Essa nova classe de sistemas computacionais, reativos ao contexto, abre perspectivas para o desenvolvimento de aplicações mais ricas, elaboradas e complexas, que exploram a natureza dinâmica das modernas infraestruturas computacionais e a mobilidade do usuário [Caceres and Friday 2012].

A revisão de literatura aponta que a construção do suporte à consciência do contexto para as aplicações ubíquas apresenta diversos desafios de pesquisa, dentre eles: (i) a aquisição do contexto a partir de fontes heterogêneas e distribuídas; (ii) o processamento das informações de contexto adquiridas e a respectiva atuação sobre o meio físico; e (iii) a disseminação do contexto a consumidores interessados de forma distribuída e no momento oportuno [Bettini et al. 2010] [Bellavista et al. 2012].

A proposta do EXEHDA-UC tem como objetivo central contribuir com o Subsistema de Adaptação e Reconhecimento de Contexto do *middleware* EXEHDA (*Execution Environment for Highly Distributed Applications*) [Lopes et al. 2012], através da

concepção de uma arquitetura de software que, de forma distribuída, ofereça às aplicações suporte às etapas de aquisição, armazenamento e processamento de informações contextuais e os decorrentes procedimentos de atuação sobre o meio.

As aplicações do ambiente gerenciado pelo EXEHDA são, por natureza, distribuídas, móveis e conscientes do contexto, estando disponíveis a partir de qualquer lugar, todo o tempo. O EXEHDA contempla na sua estrutura um núcleo mínimo e serviços carregados sob demanda. Os principais serviços fornecidos estão organizados em subsistemas relacionados a acesso ubíquo, comunicação, execução distribuída, reconhecimento do contexto e adaptação [Augustin et al. 2008].

Como premissa de concepção do EXEHDA-UC é empregada uma abordagem baseada em regras e dirigida por eventos, tanto entre os módulos da arquitetura de software como nas aplicações, para viabilizar a possibilidade de associação de regras de processamento aos contextos de interesse, as quais podem ser disparadas automaticamente quando ocorre um evento.

O artigo está organizado da seguinte forma: a seção 2 descreve a modelagem da arquitetura de software. A seção 3 apresenta um estudo de caso. A seção 4 discute os trabalhos relacionados. Por fim, a seção 5 apresenta as considerações finais.

2. EXEHDA-UC: Modelagem da Arquitetura

A concepção do EXEHDA-UC compreende dois tipos principais de servidores, que constituem o Serviço de Consciência de Contexto do EXEHDA: o Servidor de Borda, responsável pela interação com o meio através de sensores e atuadores, e o Servidor de Contexto que atua no armazenamento e processamento das informações contextuais. A arquitetura proposta provê comunicação (i) entre os Servidores de Borda e o Servidor de Contexto; (ii) entre os Servidores de Contexto localizados em diferentes células do ambiente ubíquo gerenciado pelo EXEHDA; e (iii) com outros serviços do *middleware* ou aplicações. A Figura 1 mostra uma visão da arquitetura de software do EXEHDA-UC, detalhando o Servidor de Contexto.

2.1. Servidor de Contexto

O Servidor de Contexto está organizado em seis módulos autônomos, descritos a seguir, que interoperam no provimento das funcionalidades necessárias ao Serviço de Consciência de Contexto do EXEHDA.

Coletor: provê suporte à captura das informações contextuais, coletadas pelos Servidores de Borda, considerando sensores lógicos (software) e/ou hardware.

Atuador: controla os atuadores, após ser notificado pelos outros módulos do Servidor de Contexto. Dispara no ambiente ubíquo ações que mudam o estado do meio, viabilizando o uso de serviços de consciência de contexto em aplicações de controle e automação.

Interpretador: realiza tarefas de manipulação e dedução das informações contextuais, utilizando para isto informações especificadas nos **Contextos de Interesse** das aplicações. Este módulo mantém um **Repositório de Contexto**, onde são armazenadas as informações contextuais obtidas pelo Coletor, provendo a possibilidade de registro

histórico dos contextos, o que permite a construção de regras que explorem aspectos temporais. Esses dados são utilizados pelas regras do componente **Tratador de Regras**, o qual dispara as ações pertinentes em função do estado contextual. A natureza das regras - tratamento lógico, numérico ou temporal - é uma decorrência do tipo de domínio da aplicação atendida pelo Serviço de Consciência de Contexto do EXEHDA.

Notificador: notifica o resultado do processamento contextual realizado pelo Interpretador. Recebe subscrições de todos os serviços e/ou aplicações que desejem notificações a respeito dos estados contextuais, interoperando através do módulo Gerenciador de Comunicação. Passa pelo Notificador todas as decisões de atuação provenientes do tratamento autonômico de regras de processamento contextual.

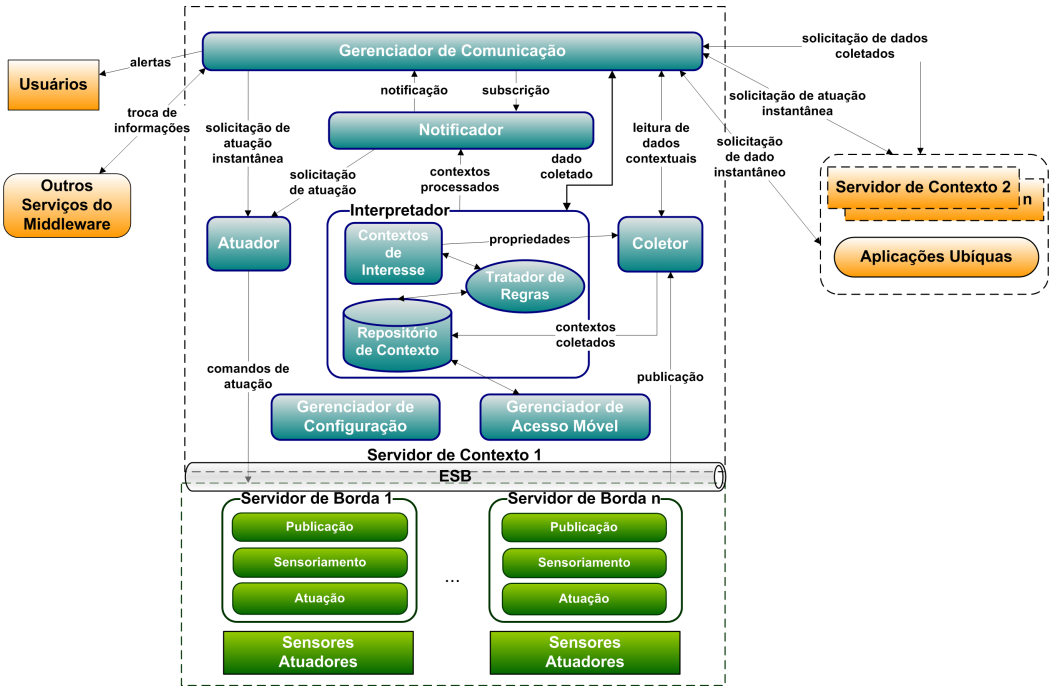

Figura 1. Visão da Arquitetura do EXEHDA-UC

Gerenciador de Comunicação: empregado por Servidores de Contexto remotos e/ou aplicações quando da solicitação de dados contextuais e/ou o disparo de atuadores. Esse módulo provê a disseminação de informações para outros serviços do *middleware*, bem como o envio de mensagens aos usuários.

Gerenciador de Configuração: permite ao usuário um gerenciamento confortável das configurações do Servidor de Contexto. O mesmo provê facilidades para que sejam especificados os diferentes aspectos dos sensores e atuadores, bem como informações dos equipamentos cujo contexto está sendo aquisitado.

Gerenciador de Acesso Móvel: provê acesso móvel ao EXEHDA-UC, possibilitando a exibição de informações contextuais e a disponibilização de alertas proativos. Particularmente, a disponibilização de alertas proativos em uma plataforma de hardware que possa acompanhar o usuário enquanto este desempenha suas atividades nos mais diferentes lugares, se mostrou um procedimento que potencializa a ubiquidade da solução de consciência de contexto disponibilizada.

2.2. Servidor de Borda

A arquitetura proposta para o Servidor de Borda contempla três módulos destinados a: (i) tratar a rede de sensores; (ii) efetuar as publicações; e (iii) tratar a rede de atuadores. Uma visão detalhada da arquitetura é mostrada na Figura 2.

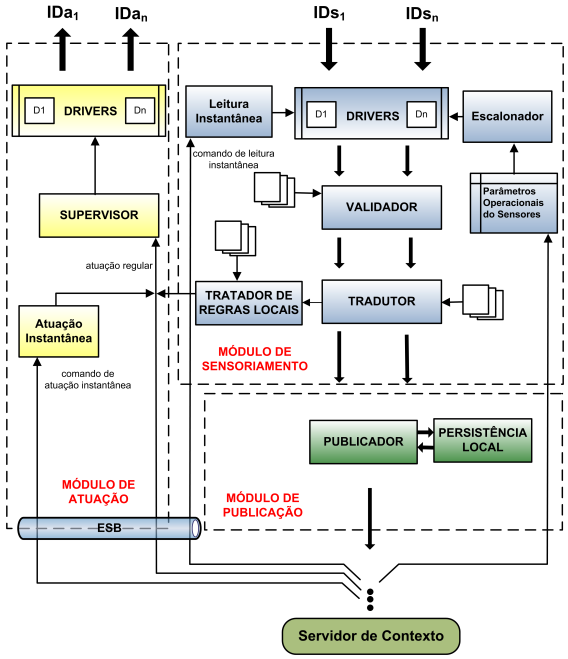

Figura 2. Detalhamento da Arquitetura do Servidor de Borda

O **Módulo de Sensoriamento** provê o tratamento de uma rede de sensores permitindo uma individualização de processamento por sensor. Este tratamento é responsável por aspectos desde gerência física (interfaces, frequência de leitura) até normalização computacional (validação, tradução) dos valores aquisitados. Também, este módulo provê funcionalidades para publicação das informações coletadas da rede de sensores no Servidor de Contexto.

O componente **Parâmetros Operacionais dos Sensores** especifica o *driver* a ser utilizado para leitura dos diferentes sensores, bem como a agenda de aquisição de dados a ser praticada. Esta agenda trata individualmente os sensores, permitindo a especificação de leituras periódicas e/ou atreladas a datas e/ou horários específicos.

O componente **Escalonador**, a partir dos Parâmetros Operacionais dos Sensores, dispara leituras considerando o interesse do usuário. O Escalonador emprega o relógio do equipamento para disparo dos seus procedimentos, sendo possível especificar a frequência com que estes disparos ocorrem.

O componente **Validador** avalia se uma determinada coleta de dados sensorados deve ser publicada ou não, empregando para isso critérios especificados pelo usuário. Esses critérios são baseados em regras, por exemplo, uma regra pode estabelecer que somente devem ser publicados valores cuja variação seja superior em 5% da leitura anterior, ou publicar valores que estão em uma determinada faixa. A regra a ser utilizada está identificada no componente Parâmetros Operacionais dos Sensores com o ID do sensor (IDs).

O componente **Tradutor** objetiva adequar o dado coletado à natureza da aplicação do usuário. Por exemplo, faixas de temperatura podem ser convertidas em descrições do tipo "Alta", "Média", "Baixa", através de uma regra procedural. Este componente contribui para otimizar os volumes de dados transferidos entre os servidores, também aumentando a legibilidade dos mesmos.

O componente **Leitura Instantânea** tem o objetivo de permitir a leitura de um determinado sensor sob demanda das aplicações a qualquer momento. As requisições de leitura podem ser provenientes de aplicações de usuários ou de Servidores de Contexto como consequência da execução de regras. Este componente emprega um barramento ESB (*Enterprise Service Bus*) para recepção assíncrona das solicitações e, a partir do ID do sensor, dispara o *driver* correspondente.

O componente **Tratador de Regras Locais** é responsável por tratar regras de contingência, com a intenção de evitar que os dispositivos envolvidos atinjam estados críticos. As mesmas atuam sobre o mecanismo de gerência da atuação ativando ou desativando atuadores. Esse componente é ativado sempre que ocorrer a leitura de um determinado sensor. Este tratador adquire elevada importância quando a comunicação com o Servidor de Contexto for interrompida por problemas de infraestrutura de rede.

O **Módulo de Publicação** é responsável por coordenar o principal fluxo de dados entre os Servidores de Borda e o Servidor de Contexto, promovendo a publicação de todos os dados coletados e garantindo uma **Persistência Local** dos mesmos nos períodos que a publicação ficar inviabilizada.

O componente **Publicador** interopera com o módulo Coletor do Servidor de Contexto, realizando as submissões dos dados coletados. As transações ocorrem sobre o protocolo HTTPS (*HyperText Transfer Protocol Secure*), o que garante que os dados serão transferidos criptografados, evitando acessos indevidos aos mesmos.

O **Módulo de Atuação** é reponsável pelo gerenciamento dos atuadores. O componente **Atuação Instantânea** é similar ao serviço de leitura instantânea dos sensores, recebendo comandos assíncronos, a qualquer momento, através um barramento ESB. Esses comandos são originários tanto do Servidor de Contexto como consequência da execução de uma regra, como de uma aplicação controlada pelo usuário.

O componente **Supervisor** aglutina comandos de atuação provenientes de três fontes distintas: atuação regular; atuação instantânea; e Tratador de Regras Locais. Uma vez recebidos os parâmetros para controle da atuação, o componente Supervisor tem autonomia para ativar o *driver* necessário para o atuador que está sendo gerenciado.

Os *drivers* dos atuadores tem objetivo similar aqueles dos sensores, ou seja, encapsulam os procedimentos específicos de cada atuador, na maioria das vezes empregando bibliotecas e/ou softwares disponibilizados por fabricantes. Esta abordagem preserva a propagação de aspectos de implementação para as camadas superiores da arquitetura do Servidor de Borda.

3. EXEHDA-UC: Estudo de Caso

Nesta seção estão resumidos os principais aspectos do estudo de caso relacionado ao projeto AMPLUS (http://amplus.ufpel.edu.br/), empregado na avaliação das funcionalidades do EXEHDA-UC. O estudo de caso contemplou tarefas referentes ao sensoriamento e a

coleta de dados contextuais, bem como o processamento e notificação das informações de contexto. Neste estudo de caso foi desenvolvida uma aplicação (vide seção 3.1) para interface Web e para dispositivos móveis, cuja usuabilidade foi avaliada pelos usuários (vide seção 3.2).

O Projeto AMPLUS foi concebido para prover serviços de consciência do contexto, permitindo um registro dos estados contextuais em que se encontram os equipamentos do Laboratório Didático de Análise de Sementes - LDAS (http://amplus.ufpel.edu.br/ldas), ao longo de todo o tempo de realização dos vários testes, e uma atuação proativa quando necessário.

Para prototipação dos mecanismos de coleta, armazenamento e processamento do contexto, bem como atuação junto ao meio foram selecionadas tecnologias escaláveis e robustas. Nesse sentido, o código dos servidores de contexto e de borda está escrito na linguagem Phyton, sendo empregado XML-RPC (*Extensible Markup Language - Remote Procedure Call*) (http://www.xmlrpc.com) para implementação do barramento ESB utilizado para interoperabilidade. O repositório de contextos emprega o gerenciador PostgreSQL para implementação das bases de dados. Os sensores e atuadores são acessados pelo protocolo 1-Wire (http://ubiq.inf.ufpel.edu.br/1-wire/).

3.1. Aplicação Desenvolvida

A aplicação desenvolvida neste estudo de caso utiliza duas abordagens, definidas em conjunto com os usuários do LDAS, uma direcionada para interface Web e outra para dispositivos móveis.

A interface Web possibilita a seleção do contexto de interesse a ser exibido, disponibilizando um relatório textual com os dados coletados pertinentes a última semana. Juntamente com este relatório é disponibilizado um menu que permite selecionar uma visualização gráfica dos dados, bem como a geração de relatórios personalizados.

O relatório gráfico (vide Figura 3) oferecido pelo sistema permite a visualização simultânea das informações de vários sensores. A seleção dos sensores é feita a partir de um menu com suporte a múltipla seleção. Também é disponibilizado um recurso de inspeção que permite a comparação dos valores em um determinado instante do tempo.

Ainda, com o intuito de promover a proatividade do Projeto AMPLUS junto a comunidade usuária, foram desenvolvidas interfaces para dois serviços públicos de comunicação: e-mail e SMS para rede celular. Estas mensagens são produzidas a partir do processamento das regras contextuais de forma autônoma pelo Servidor de Contexto.

Para atender o fato da rotina de trabalho dos laboratoristas do LDAS implicar em uma mobilidade nos diversos recintos do laboratório, foi desenvolvida uma interface de alerta visual, a qual é ativada sempre que um dispositivo estiver em um estado contextual que exija atenção. A partir deste alerta, detalhes podem ser inferidos através da interface computacional do Projeto AMPLUS.

A interface para acesso móvel foi concebida para a plataforma Android. Através da interface de abertura é possível selecionar o sensor a ser exibido, seja através de um relatório gráfico ou textual. Os relatórios gráficos e textuais oferecem a opção do usuário especificar intervalo de visualização (hora, dia, semana), sendo o ajuste do eixo vertical feito de forma automática, minimizando o emprego de rolagem de tela. Para exibição

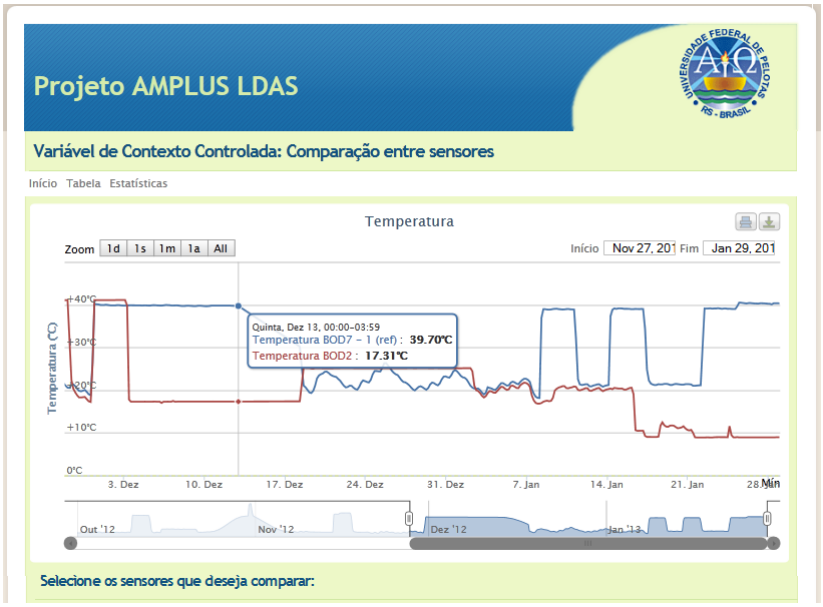

Figura 3. Projeto AMPLUS - Relatório gráfico

dos alertas foi explorada a interface disponibilizada pela plataforma Android para esta finalidade, este aspecto potencializa a integração do mecanismo de alertas às funcionalidades do *smartphone* do usuário. As interfaces correspondentes a estas funcionalidades são exibidas na Figura 4.

Figura 4. Interfaces da aplicação móvel

3.2. Avaliação de Aceitação

Esta seção apresenta um detalhamento do experimento e os resultados obtidos com a avaliação de aceitação da aplicação. O estudo envolveu 10 voluntários, entre professores, alunos e técnicos, com atividades relacionadas ao LDAS. Cada participante utilizou um dispositivo móvel e um desktop para acessar a aplicação. Após a realização de um treinamento básico, os participantes utilizaram a aplicação e responderam um questionário de avaliação, considerando a experiência de uso.

O questionário foi construído com base no Modelo de Aceitação de Tecnologia (TAM) [Yoon and Kim 2007], usando uma escala de Likert (http://psycnet.apa.org/psycinfo/1933-01885-001). Para a aceitação da aplicação

o modelo TAM considera: (i) Facilidade de uso: grau em que o usuário avalia que a aplicação pode reduzir seu esforço; e (ii) Percepção de utilidade: grau em que o usuário avalia que a aplicação pode melhorar a sua experiência.

As Tabelas 1 e 2 contêm, respectivamente, o questionário aplicado aos usuários e as respostas obtidas para facilidade de uso e percepção de utilidade. Em ambas tabelas, a primeira coluna corresponde a questão, as seguintes cinco colunas apresentam os resultados obtidos em cada escala, em graus relativos e absolutos, e a última coluna mostra a média consolidada da percentagem, variando de 0 a 5.

Tabela 1. Avaliação da Facilidade de Uso

Questão	Discordo Totalmente	Discordo Parcialmente	Indiferente	Concordo Parcialmente	Concordo Totalmente	Média
1. A aplicação é fácil de entender.	0,0%(0)	0,0%(0)	0,0%(0)	40,0%(4)	60,0%(6)	4,6
2. A aplicação é fácil de usar.	0,0%(0)	0,0%(0)	0,0%(0)	30,0%(3)	70,0%(7)	4,7
3. As opções são claras e objetivas.	0,0%(0)	0,0%(0)	10,0%(1)	20,0%(2)	70,0%(7)	4,6
4. Com pouco esforço consigo selecionar um contexto de interesse.	0,0%(0)	0,0%(0)	0,0%(0)	20,0%(2)	80,0%(8)	4,8
5. Com pouco esforço consigo acessar os relatórios gráficos.	0,0%(0)	0,0%(0)	0,0%(0)	30,0%(3)	70,0%(7)	4,7

Analisando os resultados pode-se observar que a aprovação é alta, tanto para facilidade de uso, como para percepção de utilidade. Entretanto, ocorreram resultados na escala "indiferente" nas duas últimas questões da percepção de utilidade. Isso pode ser interpretado como uma preocupação com o controle da qualidade dos experimentos desenvolvidos no LDAS, em função do uso de mecanismos autonômicos, sem a usual intervenção humana, na emissão de alertas para estados contextuais que exijam uma atuação imediata. Nesse caso, uma estratégia que pode ser adotada é intensificar os testes e validações com os usuários e iniciar uma implantação gradativa das aplicações.

Tabela 2. Avaliação da Percepção de Utilidade

Questão	Discordo Totalmente	Discordo Parcialmente	Indiferente	Concordo Parcialmente	Concordo Totalmente	Média
1. As opções apresentadas são relevantes.	0,0%(0)	0,0%(0)	0,0%(0)	30,0%(3)	70,0%(7)	4,7
2. A aplicação facilita a obtenção de dados de contexto envolvendo múltiplos sensores.	0,0%(0)	0,0%(0)	0,0%(0)	40,0%(4)	60,0%(6)	4,6
3. A aplicação facilita a minha mobilidade.	0,0%(0)	0,0%(0)	0,0%(0)	40,0%(4)	60,0%(6)	4,6
4. A aplicação facilita a atuação imediata a partir da emissão de um alerta ou mensagem.	0,0%(0)	0,0%(0)	30,0%(3)	30,0%(3)	40,0%(4)	4,1
5. Eu usaria essa aplicação no meu trabalho.	0,0%(0)	0,0%(0)	30,0%(3)	20,0%(2)	50,0%(5)	4,2

4. Trabalhos Relacionados

O estudo dos trabalhos relacionados (CARE [Agostini et al. 2009], CoCA [Ejigu et al. 2008], HiCon [Cho et al. 2008], Solar [Chen et al. 2008], WComp [Ferry et al. 2011]) foi desenvolvido considerando as premissas de concepção do

EXEHDA-UC: (1) arquitetura (distribuída ou centralizada); (2) sensoriamento (suporte a redes de sensores); (3) aquisição dos dados de contexto; (4) suporte ao tratamento de regras e; (5) suporte à atuação sobre o meio.

As arquiteturas estudadas não mantém um caráter descentralizado para todas as etapas de tratamento dos dados de contexto, o que não é oportuno para o requisito de distribuição em larga escala dos ambientes ubíquos. Por sua vez, o modelo arquitetural do EXEHDA-UC diferencia-se dos trabalhos relacionados por estar estruturado de forma distribuída, em todas as etapas de tratamento das informações de contexto, desde a aquisição até os procedimentos de atuação sobre o meio.

O EXEHDA-UC pode gerenciar redes de sensores e atuadores, tal característica é encontrada em parte nos projetos CoCA e HiCon, que têm suporte a redes de sensores. O projeto WComp, por sua vez, permite atuação sobre o meio, entretanto, não suporta o gerenciamento de redes de atuadores.

Com exceção dos projetos CARE e Solar, os demais prevêem o emprego de mecanismos específicos para aquisição do contexto, que adotam uma estratégia de separação entre a obtenção e o uso do contexto. Além de contemplar esse aspecto, o EXEHDA-UC apresenta um diferencial em relação aos projetos relacionados, que é o emprego de um caráter autônomo na aquisição dos dados de contexto, visto que estes continuam a ser obtidos pelo mecanismo, mesmo que as aplicações interessadas em seu uso estejam inoperantes.

A maioria dos projetos estudados possui suporte ao tratamento de regras, porém esta funcionalidade usualmente está restrita a algumas etapas do processamento do contexto. O EXEHDA-UC, diferencia-se destes trabalhos, por sua arquitetura de software ter sido concebida para dar suporte ao tratamento distribuído de regras personalizáveis, as quais podem estar vinculadas aos diferentes níveis de tratamento dos dados contextuais, tanto nos Servidores de Borda, como nos Servidores de Contexto.

5. Considerações Finais

O principal diferencial do EXEHDA-UC em relação aos trabalhos relacionados diz respeito a possibilidade de gerenciar a aquisição, armazenamento e processamento dos dados de contexto, de forma distribuída, independente das aplicações, em uma perspectiva autonômica baseada em regras.

No que tange à interoperabilidade entre os servidores que compõem o Serviço de Consciência de Contexto, o EXEHDA-UC contempla uma abordagem compatível com a expectativa de operação, utilizando a Internet como suporte às comunicações. Nesse sentido, o barramento ESB proposto é suportado pelo XML-RPC, o qual utiliza HTTP como protocolo físico. O *middleware* EXEHDA até então empregava para comunicações uma variante do protocolo RMI, denominada WORB [Yamin 2004], a qual não se mostra oportuna quando os equipamentos estão localizados em diferentes instituições, com requisitos particulares de segurança e acesso.

Como decorrência do emprego da Internet e de suas métricas de comunicação características, as informações contextuais que podem ser medidas ficam restritas a velocidade média dos canais de comunicação empregados. Ressalte-se que o impacto da velocidade dos canais é majoritário em relação ao desempenho de possíveis alternativas

de implementação do barramento ESB.

Dentre outros, na continuidade da pesquisa, os seguintes aspectos deverão ser considerados em trabalhos futuros: (i) explorar estudos de caso em que as regras de processamento contextual utilizem outros mecanismos de inferência de mais alto nível; e (ii) utilizar a arquitetura do EXEHDA-UC para provimento de consciência de situação.

Referências

Agostini, A., Bettini, C., and Riboni, D. (2009). Hybrid reasoning in the care middleware for context awareness. *Int. J. Web Eng. Technol.*, 5(1):3–23.

Augustin, I., Yamin, A. C., and Silva, L. C. d. (2008). Building a smart environment at large-scale with a pervasive grid middleware. In Wong, J., editor, *Grid Computing Research Progress*, volume 1, chapter 10, pages 323–344. Nova Science, New York, NY, USA.

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

Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., and Riboni, D. (2010). A survey of context modelling and reasoning techniques. *Pervasive Mob. Comput.*, 6(2):161–180.

Caceres, R. and Friday, A. (2012). Ubicomp systems at 20: Progress, opportunities, and challenges. *Pervasive Computing, IEEE*, 11(1):14–21.

Chen, G., Li, M., and Kotz, D. (2008). Data-centric middleware for context-aware pervasive computing. *Pervasive Mob. Comput.*, 4(2):216–253.

Cho, K., Hwang, I., Kang, S., Kim, B., Lee, J., Lee, S., Park, S., Song, J., and Rhee, Y. (2008). Hicon: a hierarchical context monitoring and composition framework for next-generation context-aware services. *Network, IEEE*, 22(4):34 –42.

Ejigu, D., Scuturici, M., and Brunie, L. (2008). Hybrid approch to collaborative context-aware service plataform for pervasive computing. *Journal of Computers*, 3:40–50.

Ferry, N., Hourdin, V., Lavirotte, S., Rey, G., Riveill, M., and Tigli, J.-Y. (2011). Wcomp, a middleware for ubiquitous computing, ubiquitous computing. In Babkin, E., editor, *Ubiquitous Computing*, volume 1, chapter 8, pages 171–176. InTech.

Lopes, J. L., Souza, R. S., Gusmao, M. Z., Costa, C. A., Barbosa, J. V., Yamin, A. C., and Geyer, C. R. (2012). A model for context awareness in ubicomp. In *Proceedings of the 18th Brazilian symposium on Multimedia and the web*, WebMedia '12, pages 161–168, New York, NY, USA. ACM.

Yamin, A. (2004). *Arquitetura para um Ambiente de Grade Computacional Direcionado às Aplicações Distribuídas, Móveis e Conscientes do Contexto da Computação Pervasiva*. Tese (doutorado em ciência da computação), Instituto de Informática, UFRGS, Porto Alegre, RS.

Yoon, C. and Kim, S. (2007). Convenience and tam in a ubiquitous computing environment: The case of wireless lan. *Electron. Commer. Rec. Appl.*, 6(1):102–112.