# Uma Abordagem Autonômica Baseada em Regras para Consciência de Situação na Computação Ubíqua

João Lopes*, Rodrigo Souza*, Alexandre Souza†, Patricia Davet†, Ana Pernas†, Adenauer Yamin† and Cláudio Geyer*

*PPGC/UFRGS, Porto Alegre-RS, Brasil - {jlblopes, rssouza, geyer}@inf.ufrgs.br
†PPGC/CDTec/UFPel, Pelotas-RS, Brasil - {arrsouza, pdavet, marilza, adenauer}@inf.ufpel.edu.br

***Resumo*—Um dos principais desafios de pesquisa da Computação Ubíqua (UbiComp) diz respeito a necessidade das aplicações terem consciência do contexto situacional e, quando apropriado, responder de forma proativa a mudanças no mesmo. Este artigo apresenta uma arquitetura para consciência de situação, denominada EXEHDA-SA (*Execution Environment for Highly Distributed Applications-Situation Awareness*). Considera-se como principal contribuição deste trabalho a concepção de uma arquitetura para aquisição, processamento e disseminação de informações contextuais, de forma distribuída, independente das aplicações, em uma perspectiva autonômica e baseada em regras. Para avaliar as funcionalidades do EXEHDA-SA é apresentado um estudo de caso que destaca a prototipação e testes realizados.**

## I. Introdução

A ubiquidade e a transparência constituem-se nas premissas fundamentais da Ubicomp. Estas premissas geram uma série de desafios relacionados ao acesso do usuário a seu ambiente computacional, em qualquer lugar, todo o tempo, com qualquer dispositivo, de forma não intrusiva, mantendo o foco do usuário em suas atividades. Nesta perspectiva, o sistema computacional deve interagir de forma autonômica, não importando onde esteja o usuário, constituindo um ambiente altamente distribuído, heterogêneo, dinâmico e móvel [1].

Na Computação Ubíqua as aplicações, quando necessário, podem adaptar-se a uma nova situação de seus contextos de interesse. Essa classe de sistemas computacionais, conscientes do contexto situacional, abre perspectivas para o desenvolvimento de aplicações mais ricas, elaboradas e complexas, que exploram a natureza dinâmica das infraestruturas computacionais e a mobilidade do usuário [2].

A consciência de situação, na perspectiva das aplicações ubíquas, possui três níveis: (i) percepção das modificações no ambiente, que sejam de seu interesse; (ii) compreensão do contexto situacional corrente; e (iii) projeção, em um futuro próximo, do estado de seus contextos de interesse. Com isso, uma situação corresponde a uma visão de alto nível e abrangente do contexto, que pode ser utilizada pelas aplicações em seu processo de adaptação. Esta visão é decorrente da construção de contextos compostos a partir de dados obtidos por diferentes sensores distribuídos no ambiente ubíquo [3].

A revisão de literatura aponta que a construção do suporte à consciência de situação para as aplicações ubíquas apresenta diversos desafios de pesquisa, dentre eles: (i) a coleta 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 [4] [5] [6].

As aplicações do ambiente gerenciado pelo *middleware* EXEHDA devem ser 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 [7].

A proposta do EXEHDA-SA tem como objetivo central contribuir com o Subsistema de Reconhecimento de Contexto e Adaptação do *middleware* EXEHDA, através da concepção de uma arquitetura para consciência de situação, que ofereça às aplicações suporte às etapas de coleta, processamento e disseminação de informações contextuais, bem como aos procedimentos de atuação sobre o meio físico.

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

## II. Concepção da Arquitetura

Como premissa de concepção do EXEHDA-SA é empregada uma abordagem autonômica, 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.

A concepção do EXEHDA-SA compreende dois tipos principais de servidores: o Servidor de Borda, responsável pela interação com o meio físico através de sensores e atuadores, e o Servidor de Contexto que atua no armazenamento, processamento e disseminação das informações de contexto situacional.

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. Uma visão geral desta arquitetura é apresentada na Figura 1, na qual os componentes da mesma são mapeados sobre o ambiente ubíquo gerenciado pelo EXEHDA.

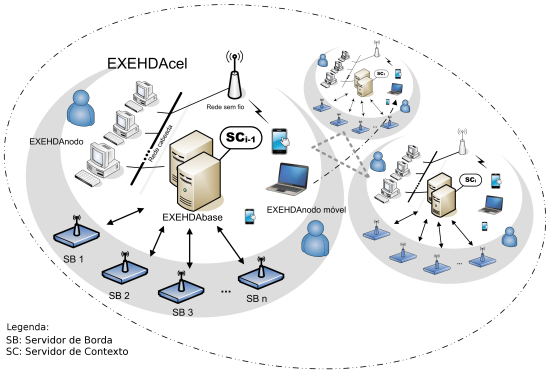


Figura 1. Ambiente Ubíquo

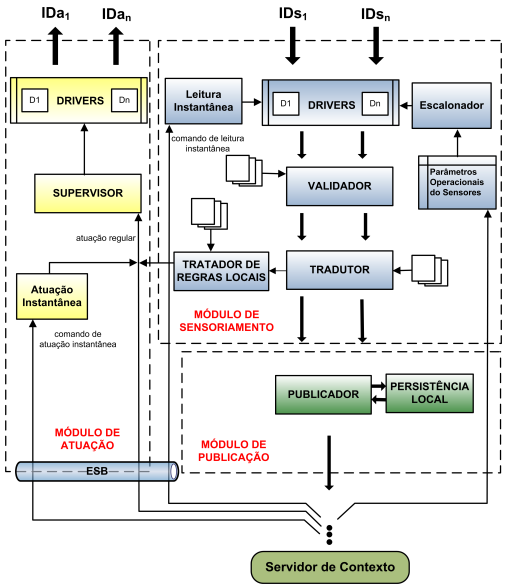


Figura 2. Servidor de Borda

A arquitetura concebida tem a capacidade de adquirir de forma autonômica informações contextuais, bem como permitir atuação remota sobre o meio físico. No ambiente ubíquo, as diferentes informações de contexto estão espalhadas. Assim, para que o usuário tenha acesso as mesmas, é necessário que elas sejam coletadas através de dispositivos sensores geograficamente distribuídos. Estas informações também devem ser registradas para seu futuro processamento tanto pelo *middleware*, como pelas aplicações.

A descrição do EXEHDA-SA, apresentada nas próximas seções, está organizada a partir de seus servidores e respectivas funcionalidades, sendo realizadas as necessárias associações entre os mesmos, e com os outros serviços do *middleware* EXEHDA.

### A. Servidor de Borda

A arquitetura proposta para o Servidor de Borda contempla três módulos destinados a tratar a rede de sensores e de atuadores, bem como efetuar as publicações. Uma visão da arquitetuta do Servidor de Borda é mostrada na Figura 2.

#### *Módulo de Sensoriamento*

O **Módulo de Sensoriamento** provê o tratamento de uma rede de sensores permitindo a individualização de processamento por sensor. Este tratamento é responsável por aspectos desde a gerência física (interfaces, frequência de leitura) até a normalização computacional (validação, tradução) dos valores coletados. Também, este módulo provê funcionalidades para publicação das informações coletadas pela 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. Por ser um componente definido pelo usuário, o mesmo é editado, no formato de um arquivo YAML[1], no Servidor de Contexto, aproveitando as especificações disponíveis no Repositório de Contexto. Uma vez pronto, o arquivo YAML correspondente é transferido para o Servidor de Borda.

O componente **Escalonador**, a partir dos Parâmetros Operacionais dos Sensores, dispara leituras considerando o interesse do usuário. O Escalonador emprega o relógio do Servidor de Borda para disparo dos seus procedimentos, sendo possível especificar a frequência com que estes disparos ocorrem. A cada ativação é disparado um parser YAML para interpretação de qual sensor deverá ser lido e ativado o *driver* correspondente.

O *driver* do sensor é responsável pela aquisição do valor referente a grandeza física medida pelo sensor. É usual cada tipo de sensor possuir uma forma específica de acesso para obtenção dos dados. A estratégia de encapsular a operação do *driver* do sensor tem por objetivo evitar que as diferenças operacionais de cada *driver* se projetem nas outras camadas de software da arquitetura. Dentre outros aspectos este encapsulamento contribui para a manutenção (troca de sensores, atualização de *drivers* por parte do fabricante). O *driver* de cada sensor é individualizado por um código identificador, permitindo que um novo *driver* possa ser adicionado, ou um *driver* antigo seja alterado, sem a necessidade de outras especificações.

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

[1] http://www.yaml.org

no componente Parâmetros Operacionais dos Sensores, a partir do 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. 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, adquirindo elevada importância quando a comunicação com o Servidor de Contexto for interrompida por problemas de infraestrutura de rede.

*Módulo de Publicação*

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 de Aquisição do Servidor de Contexto. As submissões dos dados coletados acontecem empregando o barramento ESB do Módulo de Aquisição, de forma particularizada por aplicação. Dependendo da natureza da aplicação, como procedimento de segurança os dados são transferidos criptografados.

*Módulo de Atuação*

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, em decorrência da execução de uma regra, como de uma aplicação controlada pelo usuário. Os parâmetros empregados são o ID do atuador e os correspondentes padrões de operação (tempo de duração, potência de ativação, etc.), os quais são repassados ao componente Supervisor para tratamento.

O componente **Supervisor** aglutina comandos de atuação provenientes de três fontes distintas: (i) atuação regular; (ii) atuação instantânea; e (iii) 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.

Com o intuito de identificar comportamentos anômalos no que diz respeito à gerencia dos atuadores, o Supervisor conta com uma especificação por atuador (IDa) de quantas vezes no máximo para uma unidade de tempo é esperado que ocorram transições no estado do atuador. O objetivo com isto é identificar eventuais conflitos entre as regras no Servidor de Contexto e as regras de contingência no componente Tratador de Regras Locais, ou ainda, inconformidades entre comandos de atuação instantânea disparados pelo usuário e as regras ativas nos servidores. Também, é este componente que trata os parâmetros de acionamento dos atuadores, implementando através de drivers procedimentos de ativação e desativação, de regulação de potência operacional, tempo de validade do procedimento de atuação, dentre outros.

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.

### *B. Servidor de Contexto*

Os módulos que constituem o Servidor de Contexto interoperam no provimento das funcionalidades para suporte à consciência de situação no EXEHDA. Estes módulos, descritos a seguir, são responsáveis pelas etapas que abrangem desde a aquisição do contexto até o momento em que este é armazenado e/ou repassado a quem o solicitou.

Uma visão geral da arquitetura do Servidor de Contexto é apresentada na Figura 3, na qual é caracterizada a relação com os Servidores de Borda, outros serviços do *middleware*, outros Servidores de Contexto remotos e aplicações de usuários.

**Módulo de Aquisição**: responsável por prover suporte à captura das informações contextuais, coletadas pelos Servidores de Borda considerando sensores lógicos (interfaces de software) e/ou hardware. As funcionalidades deste módulo são implementadas através de um barramento ESB, permitindo que Servidores de Borda, sempre que necessário, possam publicar os dados contextuais.

**Módulo de Atuação**: responsável pelo controle da ativação, desativação e configuração dos atuadores, após ser notificado pelos 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. De modo geral, o Módulo de Atuação é responsável por disparar no ambiente ubíquo ações que mudam o estado do meio físico, viabilizando o uso de serviços de consciência de situação em aplicações de controle e automação.

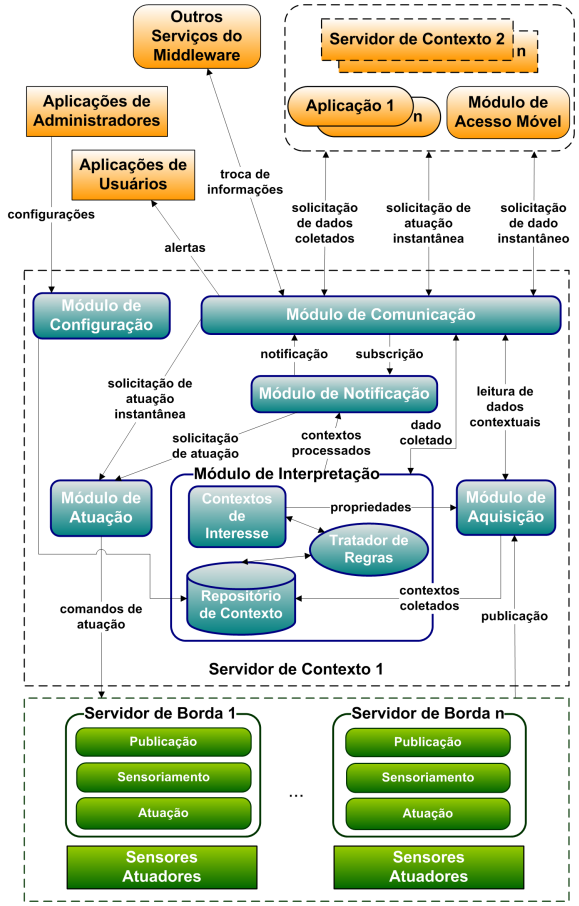


Figura 3. Servidor de Contexto

**Módulo de Interpretação**: tem como principal função realizar 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**, que atende as demandas de representação do contexto, através de um modelo relacional. Neste repositório são armazenadas as informações contextuais obtidas pelo Módulo de Aquisição, possibilitando o processamento histórico dos contextos. A estrutura do Repositório de Contexto traduz a organização da arquitetura do *middleware* EXEHDA, contemplando as relações entre as aplicações, componentes, sensores, ambientes e os contextos de interesse. O repositório armazena ainda os dados de configuração da arquitetura e as publicações provenientes dos sensores existentes no ambiente ubíquo monitorado. Esses dados são utilizados pelas regras do **Tratador de Regras**, disparando as ações pertinentes em função do estado do contexto situacional. A natureza das regras - tratamento lógico, numérico ou temporal - é uma decorrência do tipo de domínio da aplicação atendida pelo EXEHDA-SA.

**Módulo de Notificação**: responsável por notificar o resultado do processamento contextual realizado pelo Módulo de Interpretação. Esse módulo opera recebendo subscrições dos serviços e/ou aplicações que desejem notificações a respeito dos estados contextuais, interoperando através do Módulo de Comunicação. Passam pelo Módulo de Notificação todas as decisões de atuação provenientes do tratamento autonômico de regras de processamento contextual.

**Módulo de Comunicação**: utilizado por Servidores de Contexto remotos e/ou aplicações para 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. Recebe as requisições através de um barramento ESB e interopera com outros componentes da arquitetura empregando mensagens próprias, bem como com os usuários utilizando protocolos públicos direcionados ao envio de mensagens através da rede celular (SMS - *Short Message Service*), do Google Talk, e e-mails. O módulo contempla um repositório responsável por armazenar informações necessárias para o envio de alertas.

**Módulo de Configuração**: tem por objetivo permitir aos usuários administradores um gerenciamento das configurações do Servidor de Contexto. Este módulo provê funcionalidades para que sejam especificados os diferentes aspectos dos sensores e atuadores, bem como as informações dos equipamentos cujo contexto está sendo coletado.

### *Módulo de Acesso Móvel*

A disponibilização de alertas proativos em uma plataforma de hardware que possa acompanhar o usuário, enquanto este desempenha suas atividades em diferentes lugares, se mostra um procedimento que potencializa a ubiquidade da solução de consciência de situação disponibilizada. Nesse sentido, foi concebido um módulo para prover acesso móvel ao EXEHDA-SA, o qual está organizado em dois blocos (vide Figura 4): (bloco A) exibição de informações contextuais; e (bloco B) alertas proativos.

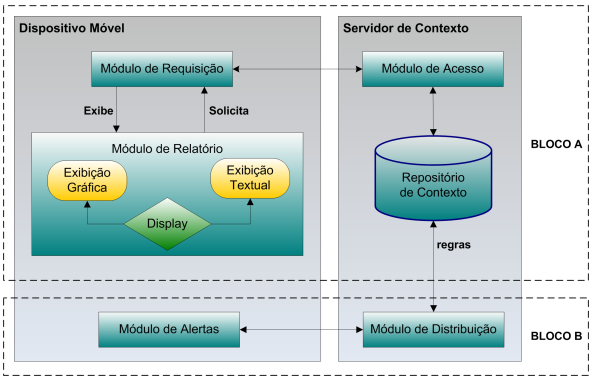


Figura 4. Módulo de Acesso Móvel

### *Bloco A: Exibição de Informações Contextuais*

Esse bloco é executado no dispositivo móvel e disponibiliza ao usuário relatórios, tanto gráficos como textuais, acerca das informações contextuais de seu interesse. A comunicação do bloco com o Servidor de Contexto acontece empregando um protocolo de dois passos. No primeiro é feita uma inspeção das informações de contexto

que estão sendo tratadas. No segundo são solicitados dados específicos para determinada informação contextual em um intervalo de tempo. As funcionalidades deste bloco foram organizadas em três módulos, conforme descrito a seguir.

**Módulo de Relatório**: trata da exibição das informações contextuais no dispositivo móvel. As principais operações disponibilizadas através deste módulo são: (i) requisição da lista de informações contextuais tratadas pelo Servidor de Contexto; (ii) seleção da informação contextual que será exibida no relatório; (iii) escolha da natureza do relatório: gráfico ou textual; e (iv) disponibilização ao usuário, uma vez exibido o relatório, de alternativas para personalização do período dos dados mostrados.

**Módulo de Acesso**: consiste de um serviço que permanece em execução junto ao Servidor de Contexto, com acesso ao Repositório de Contexto. As funcionalidades deste módulo são: (i) retornar a lista das informações contextuais gerenciadas pelo Servidor de Contexto; e (ii) retornar as informações contextuais pertinentes ao contexto de interesse do usuário, para o intervalo de tempo desejado.

**Módulo de Requisição**: tem por objetivo solicitar informações contextuais ao Módulo de Acesso, em função de solicitação do Módulo de Relatório. Este módulo contempla três funções: (i) requisitar a lista de todas as informações contextuais tratadas pelo Servidor de Contexto; (ii) requisitar uma determinada informação contextual desejada pelo usuário, no intervalo de tempo especificado; (iii) enviar esta solicitação ao Módulo de Acesso através do barramento ESB do mesmo; e (iv) receber e interpretar as informações contextuais provenientes do Módulo de Acesso, disponibilizando-as ao Módulo de Relatório.

*Bloco B: Tratamento de Alertas Proativos*

Esse bloco tem por objetivo enviar notificações proativas ao usuário, quando da ocorrência de eventos de seu interesse. O bloco tem suas funcionalidades organizadas em dois módulos, descritos a seguir.

**Módulo de Alertas**: executa no dispositivo móvel, disponibilizando alertas aos usuários. Para tanto, é empregado mecanismo nativo de notificação da plataforma móvel. Esta opção tem como característica propiciar ao usuário um gerenciamento integrado da natureza dos alertas praticados pelo seu dispositivo móvel. As principais funções deste módulo são: (i) recuperar no intervalo de tempo especificado pelo usuário os alertas junto ao Módulo de Distribuição; e (ii) disponibilizar ao usuário estes alertas na área de notificação do dispositivo móvel.

**Módulo de Distribuição**: executa no mesmo equipamento do Servidor de Contexto, em regime de operação ininterrupta. As principais funções deste módulo são: (i) receber os alertas para os dispositivos móveis produzidos pelo Tratador de Regras do Servidor de Contexto; e (ii) disponibilizar aos dispositivos móveis os alertas. Este módulo opera mantendo os alertas produzidos pelas diferentes regras contextuais, tratadas no Módulo de Interpretação do Servidor de Contexto. Seu acesso pelos dispositivos móveis ocorre através do Módulo de Comunicação, o qual através de um barramento ESB, propicia acesso às diferentes funcionalidades do Servidor de Contexto.

## III. Estudo de Caso

As funcionalidades propostas para o EXEHDA-SA são avaliadas através de um estudo de caso, relacionado ao projeto AMPLUS[2], desenvolvido no Laboratório Didático de Análise de Sementes (LDAS/FAEM/UFPel).

A principal finalidade da análise de sementes é a de determinar a qualidade de um lote e, consequentemente, o seu valor para a semeadura. Para que os objetivos das análises sejam atingidos é necessário que os laboratórios possuam equipamentos adequados e sigam métodos e procedimentos uniformes [8]. Nesse sentido, os serviços de consciência de situação do EXEHDA-SA possibilitam o registro dos estados contextuais em que se encontram os equipamentos do LDAS, ao longo de todo o tempo de realização dos vários testes de análise de sementes, bem como uma atuação proativa quando necessário.

O estudo de caso contempla as tarefas referentes ao sensoriamento e a coleta de dados contextuais, bem como o processamento e a notificação das informações de contexto. Neste estudo de caso foi desenvolvida uma aplicação (vide seção III-A) para interface Web e para dispositivos móveis, cuja usabilidade foi avaliada pelos usuários (vide seção III-B).

O código dos servidores de contexto e de borda está escrito na linguagem Phyton, sendo empregado XML-RPC (*Extensible Markup Language - Remote Procedure Call*)[3] para implementação do barramento ESB utilizado para interoperabilidade. O Repositório de Contexto emprega o gerenciador PostgreSQL para implementação das bases de dados. Os sensores e atuadores se comunicam pelo protocolo 1-Wire[4].

### A. 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 na 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 5) 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

[2]http://amplus.ufpel.edu.br
[3]http://www.xmlrpc.com
[4]http://ubiq.inf.ufpel.edu.br/1-wire/

comparação dos valores em um determinado instante do tempo. A janela de tempo dos dados que estão sendo visualizados pode ser definida pelo usuário através da mesma interface gráfica que exibe os valores sensorados.

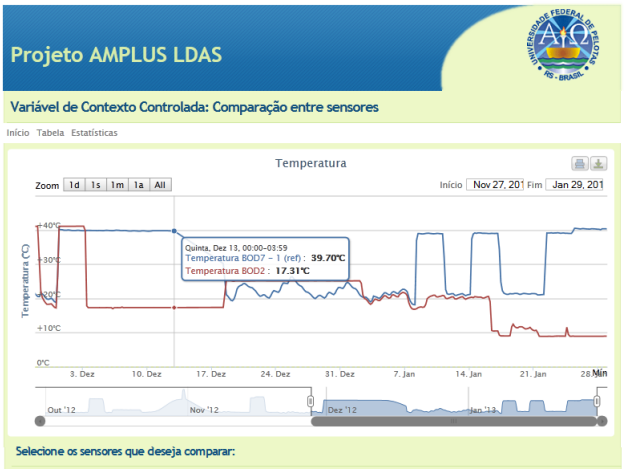


Figura 5. Relatório Gráfico

Para melhor suportar as necessidades de dados para as pesquisas realizadas no LDAS foi concebida uma funcionalidade que realiza o cruzamento de dados contextuais envolvendo múltiplos sensores a partir de diferentes regras. Toda a manipulação é feita através da interface Web com facilidades para adição, remoção e edição de regras e seus parâmetros, como mostra a Figura 6.

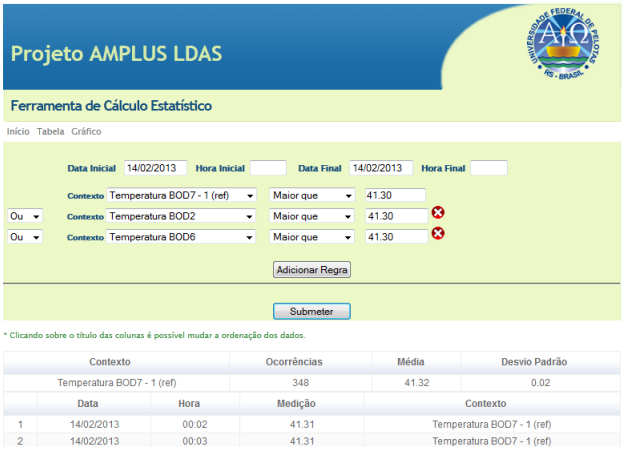


Figura 6. Cruzamento de Informações Contextuais

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 Internet 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 voltada a dispositivos móveis 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. Estes relatórios 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 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 7.

### B. *Avaliação de Aceitação*

Esta seção apresenta o 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) [9]. 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 I e II 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.

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, para a 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.

## IV. Trabalhos Relacionados

O estudo dos trabalhos relacionados (CARE [10], CoCA [11], HiCon [12], Solar [13], WComp [14]) foi de-


Figura 7. Interfaces da Aplicação Móvel

Tabela I
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 |

Tabela II
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 |

senvolvido considerando o objetivo central do EXEHDA-SA de prover suporte à consciência de situação, bem como suas principais premissas de concepção: (i) arquitetura distribuída; (ii) suporte a redes de sensores e de atuadores; (iii) aquisição autonômica dos dados de contexto; (iv) suporte ao tratamento de regras; e (v) suporte à atuação distribuída sobre o meio físico.

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-SA 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 físico.

O EXEHDA-SA pode gerenciar redes de sensores e de atuadores, otimizando o gerenciamento tanto da aquisição dos dados de contexto a partir de vários tipos de sensores, usual nos ambientes computacionais para provimento de aplicações ubíquas, como da atuação distribuída sobre o meio físico. 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 físico, 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. Estes mecanismos adotam uma estratégia de separação entre a obtenção e o uso do contexto. Essa estratégia é um dos aspectos centrais para a concepção de arquiteturas de suporte ao processamento do contexto. O EXEHDA-SA também obtém os dados contextuais de forma separada das aplicações que os utilizam. Por outro lado, o EXEHDA-SA diferencia-se dos projetos relacionados, em função do emprego de um caráter autonômico 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 usual-

mente está restrita a algumas etapas do processamento do contexto, principalmente à interpretação dos dados contextuais. O EXEHDA-SA 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.

## V. Considerações Finais

Na perspectiva de reduzir o esforço de desenvolvimento de aplicações ubíquas no EXEHDA, as condições de contexto são proativamente monitoradas pelo EXEHDA-SA, fazendo com que o desenvolvedor fique desobrigado de gerenciar aspectos como coleta, processamento e armazenamento de dados contextuais. No EXEHDA-SA é empregada uma estratégia colaborativa entre aplicação e *middleware*, através da qual é facultado ao programador individualizar regras para reger o comportamento de componentes que constituem o software da aplicação.

O EXEHDA-SA, quando comparado com outros trabalhos da área, segue a tendência de prover uma abordagem distribuída na aquisição de contexto e/ou atuação, com o adicional de aproximar da rede de sensores uma infraestrutura com capacidade autonômica de tratamento dos dados contextuais.

Potencializando a abordagem distribuída no tratamento das informações de contexto, o EXEHDA-SA suporta o conceito de rede de sensores e de atuadores, o que além de contribuir com aspectos de modularidade do desenvolvimento do software necessário, também contribui para a organização dos procedimentos de criação e manutenção das redes envolvidas.

Outra característica relevante do EXEHDA-SA, quando comparado com os trabalhos relacionados, é que o mesmo gerencia a aquisição dos dados de contexto, seu processamento e armazenamento de forma independente das aplicações, em uma perspectiva autonômica baseada em regras. O emprego de regras em diferentes pontos da arquitetura distribuída do EXEHDA-SA constitui um diferencial significativo.

As prototipações realizadas até o momento vêm apontando resultados promissores, tanto no que tange à concepção do modelo arquitetural, como às tecnologias utilizadas. Também, as avaliações de usabilidade dos usuários do projeto AMPLUS têm sido positivas e têm trazido retornos para a consolidação do EXEHDA-SA.

Dentre outros aspectos, na continuidade da pesquisa, pretende-se explorar estudos de caso que tratem a projeção de estados futuros dos contextos de interesse, abrangendo todos os níveis da consciência de situação.

## Referências


[1] R. Caceres and A. Friday, "Ubicomp systems at 20: Progress, opportunities, and challenges," *Pervasive Computing, IEEE*, vol. 11, no. 1, pp. 14–21, 2012.

[2] J. Krumm, *Ubiquitous Computing Fundamentals*, 1st ed. Chapman & Hall/CRC, 2010.

[3] J. Ye, S. Dobson, and S. McKeever, "Review: Situation identification techniques in pervasive computing: A review," *Pervasive Mob. Comput.*, vol. 8, no. 1, pp. 36–66, Feb. 2012.

[4] C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan, and D. Riboni, "A survey of context modelling and reasoning techniques," *Pervasive Mob. Comput.*, vol. 6, no. 2, pp. 161–180, Apr. 2010.

[5] P. Bellavista, A. Corradi, M. Fanelli, and L. Foschini, "A survey of context data distribution for mobile ubiquitous systems," *ACM Comput. Surv.*, vol. 44, no. 4, pp. 24:1–24:45, Sep. 2012.

[6] T. Silva, C. Celes, V. Mota, and A. Loureiro, "Overview of ubicomp research based on scientic publications. in:," in *Simpósio Brasileiro de Computação Ubíqua e Pervasiva*. Curitiba, PR: SBC, 2012.

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

[8] ISTA, "Seed testing international," No. 144, October 2012.

[9] C. Yoon and S. Kim, "Convenience and tam in a ubiquitous computing environment: The case of wireless lan," *Electron. Commer. Rec. Appl.*, vol. 6, no. 1, pp. 102–112, Jan. 2007.

[10] A. Agostini, C. Bettini, and D. Riboni, "Hybrid reasoning in the care middleware for context awareness," *Int. J. Web Eng. Technol.*, vol. 5, no. 1, pp. 3–23, May 2009.

[11] D. Ejigu, M. Scuturici, and L. Brunie, "Hybrid approch to collaborative context-aware service plataform for pervasive computing," *Journal of Computers*, vol. 3, pp. 40–50, 2008.

[12] K. Cho, I. Hwang, S. Kang, B. Kim, J. Lee, S. Lee, S. Park, J. Song, and Y. Rhee, "Hicon: a hierarchical context monitoring and composition framework for next-generation context-aware services," *Network, IEEE*, vol. 22, no. 4, pp. 34 –42, july-aug. 2008.

[13] G. Chen, M. Li, and D. Kotz, "Data-centric middleware for context-aware pervasive computing," *Pervasive Mob. Comput.*, vol. 4, no. 2, pp. 216–253, Apr. 2008.

[14] N. Ferry, V. Hourdin, S. Lavirotte, G. Rey, M. Riveill, and J.-Y. Tigli, "Wcomp, a middleware for ubiquitous computing," in *Ubiquitous Computing*, E. Babkin, Ed. InTech, 2011, vol. 1, ch. 8, pp. 171–176.