# Middleware de Integração de Sensores para o Desenvolvimento de Aplicações Móveis Sensíveis ao Contexto

**Marcio Pereira de Sá , Vagner Sacramento ,**
**Ricardo Couto A. da Rocha , Leonardo Mayer Delfiaco , Bruccy Mendes Sodré**

[1]Instituto de Informática – Universidade Federal de Goiás (UFG)
Caixa Postal 131 – 74.001-970 – Goiânia – GO – Brazil

***Abstract.*** *Middleware platforms for context provisioning are important because they facilitate the development of context-aware applications. However, middleware systems generally do not address issues such as the complexity in development and reuse of sensor modules that collect contextual information of heterogeneous sensors. This work proposes a middleware platform for context provisioning on mobile devices, named ConBus (**Con**text **Bus**) that implements reuse, deployment and dynamic activation strategies for sensor modules.*

***Resumo.*** *Plataformas de middleware para provisão de contexto são importantes por facilitar o desenvolvimento de aplicações sensíveis ao contexto. Entretanto, em geral, os sistemas de middleware disponíveis não endereçam questões como a complexidade envolvida no desenvolvimento e reutilização de módulos de sensoriamento que realizam a coleta da informação contextual de sensores heterogêneos. Com o propósito de lidar com tais desafios, este trabalho propõe uma plataforma de middleware de provisão de contexto para dispositivos móveis, denominada ConBus (**Con**text **Bus**), que implementa estratégias de reutilização, implantação e ativação dinâmica de módulos de sensoriamento.*

## 1. Introdução

A computação ubíqua, idealizada por Mark Weiser [Weiser 1991], ainda na década de 1980, caracteriza-se por permitir que seus usuários possam ter acesso a serviços computacionais praticamente em qualquer lugar e a qualquer momento. Para se alcançar tais objetivos, é importante que as novas aplicações percebam e reajam a mudanças do ambiente no qual estão inseridas como, por exemplo, alterações na localização de um usuário, na temperatura de uma determinada sala e no estado dos recursos computacionais de um dispositivo. Todas essas informações são denominadas informações de contexto e são providas, direta ou indiretamente, por sensores. Porém, desenvolver tais aplicações ainda é uma atividade complexa por causa da imensa quantidade de informações contextuais e de tecnologias de sensoriamento.

Nesse cenário, sistemas de *middleware* assumem a responsabilidade de intermediar a comunicação entre as aplicações e os sensores que são as fontes de informações contextuais. Essa responsabilidade envolve diversos serviços como protocolo de comunicação com sensores heterogêneos, comunicação assíncrona, manutenção de modelos de contexto de alto nível, inferência de informações contextuais, modelos de adaptação baseada em contexto para aplicações e a incorporação dos dados provindos dos sensores. Neste trabalho, o termo *módulo de sensoriamento* é empregado para descrever o componente de software que se conecta a um sensor e coleta o dado de contexto propriamente dito.

Apesar de serem muito importantes para a construção de aplicações móveis sensíveis ao contexto, o desenvolvimento de plataformas de *middleware* para a provisão de contexto também é uma tarefa muito complexa, especialmente com relação à integração de módulos de sensoriamento a tais infraestruturas. Isto ocorre devido a vários fatores, como: *i)* a complexidade inerente ao desenvolvimento dos módulos de sensoriamento, que usualmente envolvem chamadas de baixo nível ao sistema operacional ou exigem a implementação de protocolos de comunicação para acesso a sensores remotos; e *ii)* dificuldades de reutilização de tais módulos e o gerenciamento do ciclo de vida desses componentes.

Com o propósito de lidar com tais desafios, este trabalho propõe uma arquitetura de *middleware* de provisão de contexto para dispositivos móveis, denominada ConBus (***Con****text* ***Bus***), que implementa estratégias de desenvolvimento, reutilização, implantação e ativação dinâmica de módulos de sensoriamento, fazendo uso racional dos recursos computacionais do dispositivo. A implementação atual é baseada na plataforma Android.

Este artigo está organizado da seguinte forma. A Seção 2 discute o problema da integração de sensores em sistemas de *middleware* para provisão de contexto. A Seção 3 apresenta a arquitetura do ConBus. A Seção 4 apresenta estudos de caso de implementação de duas aplicações baseadas no *middleware* implementado e discute como a abordagem de integração de sensores facilitou o desenvolvimento da aplicação. A Seção 5 discute trabalhos na literatura relacionados com a abordagem discutida neste artigo. A Seção 6 apresenta as conclusões deste trabalho e futuras direções de pesquisa.

## 2. Integração de Sensores a Sistemas de Middleware de Provisão de Contexto

Sistemas de *middleware* têm importância fundamental para computação sensível ao contexto por promoverem o desacoplamento entre aplicações sensíveis ao contexto e as fontes de informação contextual, além de oferecerem transparência dos mecanismos de provisão de contexto. Diversos sistemas de *middleware* têm sido propostos, desde o trabalho original do ContextToolkit [Dey et al. 2001], passando por trabalhos focados em aspectos específicos como privacidade [Hong and Landay 2004], modelagem complexa de informações contextuais [Chen 2005], cenários particulares como *smart rooms*, mecanismos de adaptação de aplicações [Henricksen and Indulska 2006], execução em *smartphones* [Riva 2006] e transparência de mecanismo de provisão de contexto [Riva 2006]. Entretanto, poucos trabalhos têm focado na facilidade de integração de sensores ao *middleware* de provisão de contexto. Segundo o modelo de camadas proposto por Henricksen e Indulska [Henricksen and Indulska 2006], essa integração é papel implementado nas camadas de recepção (*reception*) e coleta (*gathering*) de contexto.

Os desafios para integração de sensores são especialmente interessantes em sistemas de *middleware* projetados para dispositivos móveis. Em tais dispositivos, a diversidade de tecnologias de sensoriamento motivam a implementação de cenários mais amplos de computação sensível ao contexto. Por exemplo, a maioria dos *smartphones* vendidos atualmente já oferecem sensores como GPS, acelerômetro e bússola digital. Além disso, a mobilidade provida por tais dispositivos permite a elaboração de cenários dinâmicos, nem sempre previsíveis, nos quais uma mudança de localização pode ocasionar o acesso a outro sensor remoto responsável por uma mesma informação contextual (e.g. sensor de temperatura do ambiente). Por outro lado, a limitação de recursos computacionais

exige a adoção de estratégias para minimizar o uso de memória, CPU e comunicação pela rede. De fato, há vários aspectos que dificultam a integração de sensores em sistemas de *middleware* para provisão de contexto em dispositivos móveis e o desenvolvimento de módulos de sensoriamento nesses ambientes. Um problema comum às arquiteturas é a *dificuldade de construção e reutilização dos módulos de sensoriamento* devido à falta de mecanismos que facilitem o desenvolvimento, a distribuição e a manutenção dos módulos. Tais mecanismos são fundamentais para incorporação de módulos de sensoriamento associados a sensores externos ou não diretamente suportados pela plataforma. Além disso, devido à carência de recursos computacionais nos dispositivos móveis, como memória, processamento e energia, o *gerenciamento do ciclo de vida de módulos de sensoriamento* torna-se um dos grandes desafios para as arquiteturas de *middleware* para dispositivos móveis.

## 3. Arquitetura do Middleware ConBus

O ConBus é uma infraestrutura desenvolvida para prover informações de contexto a aplicações móveis. Através de seu projeto arquitetural, baseado na interação entre três camadas, ilustradas na Figura 1: *i)* Camada de Aplicação; *ii) Middleware* ConBus; e *iii)* Camada de Sensores, essa plataforma de *middleware* oferece uma arquitetura flexível em relação ao gerenciamento dos módulos de sensoriamento de contexto acoplados a ela. Por isso, o ConBus é responsável por gerenciar o ciclo de vida dos módulos de sensoriamento integrados à sua implementação, oferecer um *framework* de integração de novos módulos de sensoriamento (*ConUnit framework*), e prover às aplicações móveis sensíveis ao contexto uma interface uniforme que padroniza o acesso aos dados de contexto providos por diferentes sensores integrados à arquitetura através dos módulos de sensoriamento.

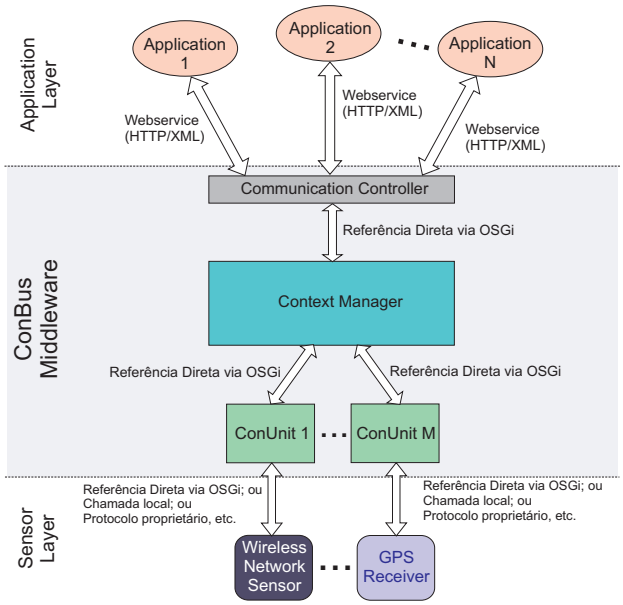


**Figura 1. Visão Geral da Arquitetura ConBus**

A Camada de Aplicação (*Application Layer*) compreende todas as aplicações sensíveis ao contexto que utilizam as informações contextuais fornecidas por uma única instância do *middleware* ConBus. As aplicações podem acessar as informações providas

pelos sensores através da *API das Aplicações* oferecida pelo *middleware*. Essa API disponibiliza os comandos através dos quais as informações de contexto podem ser requisitadas pelas aplicações, estabelecendo o formato, a sintaxe e a codificação tanto das requisições quanto das respostas, bem como os tipos de erros mais comuns para cada requisição feita ao *middleware*. A Camada de sensores (*Sensor Layer*), ilustrada na Figura 1, é constituída por sensores que fornecem informações contextuais ao *middleware ConBus*. Um sensor pode ser local ou remoto em relação ao dispositivo móvel.

### 3.1. *Middleware* ConBus

O *middleware ConBus* é a camada intermediária, na arquitetura proposta, e tem a função de gerenciar todas as informações contextuais, coletadas pelos sensores que constituem a *Camada de Sensores* e que, posteriormente, são fornecidas a todas as aplicações sensíveis ao contexto. O *framework OSGi* [Alliance. 2009] é o componente que permite o acoplamento, tanto estático quanto dinâmico, de todas as fontes de contexto junto ao *middleware* ConBus e também a interligação de todos os outros componentes internos do *middleware*. Dessa forma, o *OSGi* atua como um barramento de comunicação que permite ao *Gerenciador de Contexto*, por exemplo, controlar o uso de todos os *ConUnits* instalados, além de possibilitar o acesso dos outros componentes da arquitetura aos dados disponibilizados por essas fontes.

#### 3.1.1. ConUnit

O *framework ConUnit* (*Context Unit*) é responsável por definir como um sensor deve ser integrado ao ConBus. Através de uma instância de *ConUnit*, o ConBus poderá coletar dados de contexto de um sensor específico, de acordo com o interesse das aplicações. Em outras palavras, cada instância de *ConUnit* incorpora um determinado módulo de sensoriamento à arquitetura ConBus. Em relação ao OSGi, o *middleware* explora esse *framework* especialmente em *ConUnits*, que são implementados em *bundles OSGi*[1], como ilustra a Figura 2a. Dessa maneira, o ConBus utiliza as facilidades do *OSGi* para instalar, remover, ativar e desativar dinamicamente *ConUnits* sob a demanda de aplicações interessadas em suas informações contextuais.

Conforme ilustrado pela Figura 2b, um *ConUnit* é constituído por duas partes: uma representando a parte do *ConUnit* que é dependente de um sensor específico (*Sensor Integration*), composta pelo módulo de sensoriamento (*Sensor Module*) propriamente dito e pelo *Componente de Integração com o Sensor* (*Sensor Integration Component*) que implementa a interface *ContextDataConversion*, que permite ao desenvolvedor converter os dados coletados em um formato específico desenvolvido para o ConBus. Na parte do *ConUnit* independente de sensor (*ConBus Integration*), o *ConUnit Manager* é responsável por gerenciar todo o acesso aos dados providos pela fonte de contexto a qual o *ConUnit* está associado. Por exemplo, quando uma aplicação envia uma requisição ao ConBus desejando obter uma consulta a alguma informação contextual gerenciada por ele, essa aplicação deve informar o tipo de contexto desejado, à qual entidade (pessoa, local, objeto, etc.) esta informação está relacionada e, opcionalmente, o sensor propriamente dito

[1]Um *bundle* OSGi é a menor unidade executável em um container OSGi. Cada *bundle*, nessa arquitetura, é um componente modularizado cuja comunicação com os outros componentes do sistema se dá através de interfaces bem definidas e/ou serviços, controlados pelo *framework* OSGi.

(por exemplo, receptor GPS ou PlaceLab para coletar dados sobre a localização de um usuário). Ao chegar ao *Gerenciador de Contexto*, essa requisição é repassada ao *ConUnit Manager* da instância do *ConUnit* que faz a integração com a fonte de contexto informada. Em seguida, o *ConUnit Manager* faz uma requisição chamando o método apropriado do *Sensor Integration Component* e aguarda uma resposta, que será representada segundo o modelo de contexto do ConBus. O componente *Synchronous Controller*, também presente na *ConBus Integration* de cada *ConUnit*, apenas controla os detalhes de comunicação entre o *Gerenciador de Contexto* e o *ConUnit* correspondente.

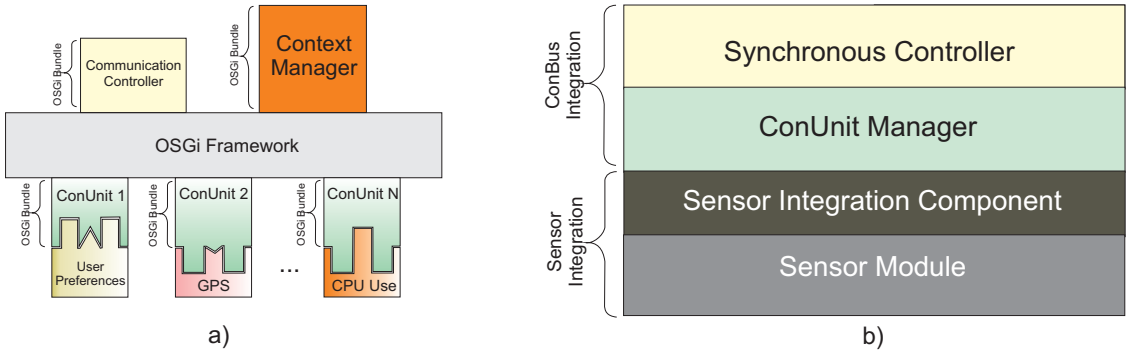


**Figura 2. a) Integração de ConUnits e outros componentes da arquitetura ConBus ao framework OSGi. b) Arquitetura de um ConUnit.**

### 3.1.2. Gerenciador de Contexto

O *Gerenciador de Contexto* (*Context Manager*) é o responsável pelo gerenciamento do ciclo de vida de cada *ConUnit* acoplado ao *middleware*. O ciclo de vida de um *ConUnit* é constituído pelas seguintes etapas: instalação, ativação, desativação e desinstalação de cada instância desse *framework*. Por isso, as tarefas do *Gerenciador de Contexto* são instalar, ativar, desativar e desinstalar dinamicamente *ConUnits* e implementar o acesso síncrono e assíncrono às informações contextuais providas por essas fontes de contexto.

O *Gerenciador de Contexto* deve ativar dinamicamente, através do *framework OSGi*, os *ConUnits* quando os dados fornecidos por eles forem requisitados pelas aplicações, desativando-os, também em tempo de execução, sempre que nenhuma aplicação ou outro componente da arquitetura estiver interessado em suas informações contextuais. Para ser ativado, cada *ConUnit* deve ser primeiramente identificado. Isso é feito através de metadados contidos no próprio *ConUnit*, como o tipo de contexto (i.e., localização, temperatura, dados da agenda de compromissos, uso de CPU, etc.) provido por ele, a qual entidade (por exemplo, LOJA_101) esta informação contextual está associada e, opcionalmente, o tipo de sensor responsável pela coleta dos dados contextuais.

Quando o *middleware* ConBus é iniciado, todos os *ConUnits* acoplados (previamente instalados) a ele encontram-se desativados. Cada módulo é carregado dinamicamente assim que recebe, pela primeira vez, uma requisição de uma aplicação através do *Gerenciador de Contexto*. Essa funcionalidade visa gerenciar melhor os recursos computacionais do dispositivo móvel, principalmente memória. A desativação acontece depois de transcorrido um certo intervalo de tempo sem que nenhuma aplicação tenha requisitado

dados provenientes desse *ConUnit*.

Além de ativar e desativar *ConUnits*, o *Gerenciador de Contexto* também implementa o acesso síncrono e assíncrono às informações contextuais providas por esses componentes, através de seus módulos de sensoriamento. No caso síncrono, o *Gerenciador de Contexto* encaminha cada requisição feita por uma aplicação ao respectivo *ConUnit*, devolvendo a resposta à aplicação correspondente. Para o caso assíncrono, as aplicações inicialmente devem registrar interesse (*subscription*) em receber uma determinada informação contextual sempre que uma condição desejada for satisfeita. Após o registro, o *Gerenciador de Contexto* ficará acessando, em intervalos de tempo determinados, o respectivo *ConUnit*, comparando os valores coletados com os registros (*subscriptions*) feitos pelas aplicações sensíveis ao contexto.

### 3.1.3. Controlador de Comunicações

O *Controlador de Comunicações* (*Communication Controller*), conforme ilustrado na Figura 1, é o componente da arquitetura responsável por receber requisições das aplicações e enviar a elas as respostas correspondentes. Sua função é receber as requisições, extrair as informações necessárias (comando enviado, bem como todos os seus parâmetros, quando houver algum), verificando possíveis erros na mensagem, enviando, em seguida, ao *Gerenciador de Contexto*, os dados necessários para que este possa processar a resposta que será enviada à aplicação que solicitou a informação.

## 4. Estudos de Caso

Esta seção discute como serviços de provisão de contexto do ConBus auxiliam o desenvolvimento de aplicações e a incorporação de módulos de sensoriamento, a partir de dois estudos de caso. Ambas as aplicações foram implementadas em Java e executadas no emulador Android.

### 4.1. SmartTravel

O SmartTravel é uma aplicação que notifica o usuário sempre que este estiver na proximidade de pontos turísticos ou de lazer previamente registrados como *pontos de interesse*, conforme ilustra a Figura 3a. Para ser executado adequadamente, o SmartTravel necessita obter as seguintes informações de contexto: localização, data e hora, dados da agenda de compromissos, tipos de mídia disponíveis no dispositivo móvel, preferências do usuário e se o usuário está ou não realizando uma chamada telefônica. Para registrar um ponto turístico ou de lazer como um ponto de interesse, o usuário deve escolher uma categoria predefinida, como “museus”, “parques”, “hotéis”, dentre outras, ou ainda cadastrar um novo ponto turístico ou de lazer. Para cada categoria, o usuário pode indicar parâmetros específicos, como “preço máximo das diárias”, para o caso de hotéis e pousadas, “horário de funcionamento”, para o caso de lojas, bares, museus, etc. Após o registro dos pontos de interesse, a aplicação está pronta para ser usada.

Uma vez iniciada, o SmartTravel começará a enviar as informações de localização juntamente com as preferências dos pontos turísticos ou de lazer previamente cadastra-

das para um servidor remoto[2]. Esse servidor permite que as instâncias do SmartTravel, localizadas nos dispositivos móveis, enviem informações sobre a localização corrente, bem como as preferências de seus usuários (os pontos turísticos previamente registrados). Após receber essas informações, o servidor remoto procurará encontrar os pontos turísticos ou de lazer que estejam nas proximidades do local onde este usuário móvel se encontra naquele momento. Caso encontre algum ponto turístico ou de lazer de interesse deste usuário, o servidor remoto enviará uma resposta ao SmartTravel com as informações sobre o ponto turístico ou de lazer encontrado. Em seguida, de posse dessas informações, a aplicação móvel irá analisar o contexto atual do usuário e do dispositivo móvel procurando encontrar a melhor maneira possível de notificá-lo (através de texto, áudio, vídeo, mapas, etc.) sobre a existência de algum ponto turístico ou de lazer de seu interesse nas proximidades. Por exemplo, ao consultar a agenda de compromissos do usuário, pode-se descobrir se o usuário tinha ou não uma reunião para aquele local e para aquele horário e, assim, postergar uma notificação, para após o seu término. As Figuras 3(b), 3(c) e 3(d) ilustram a chegada de uma notificação do SmartTravel, informando ao usuário da proximidade do ponto de referência *"Castros Park Hotel"*.

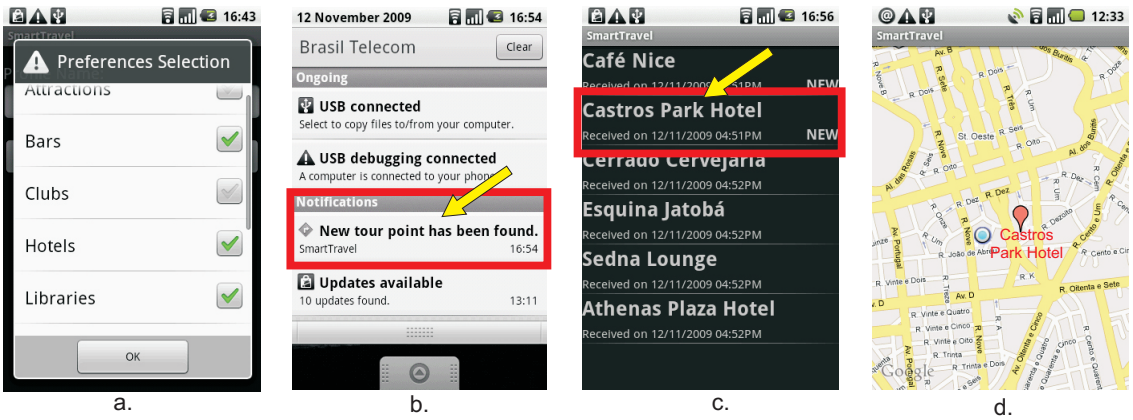


**Figura 3. Funcionamento do SmartTravel**

Do ponto de vista da provisão de contexto, todas as informações contextuais necessárias são obtidas dos seus sensores correspondentes através de um *ConUnit*. Um *ConUnit* é o componente do ConBus que integra ao *middleware* o módulo de sensoriamento responsável pela informação, i.e. para cada sensor acoplado ao ConBus, foi desenvolvido um módulo de sensoriamento específico. Dentre as informações contextuais necessárias está a informação sobre os tipos de mídia suportados pelo dispositivo móvel que é fornecida através do *Available Media ConUnit*, isto é importante para que o SmartTravel possa notificar o usuário da melhor maneira possível, visto que, mesmo sabendo que uma mensagem de áudio é mais adequada, é necessário saber se o dispositivo móvel corrente suporta tal formato de arquivo.

## 4.2. AutoRinger

O AutoRinger é uma aplicação móvel que utiliza informações de contexto, como a localização e as preferências de seus usuários para, de acordo com cada situação do dia

[2]Este servidor remoto ainda não foi implementado! Para os testes empregou-se um simulador para realizar as tarefas desse servidor.

a dia, configurar o tipo de toque de campainha mais adequado para o dispositivo móvel. Como exemplo, imagine que todo dia, ao chegar à sala de aula, um professor, manualmente, altere o tipo de toque de campainha de "normal" para "silencioso" ou "alerta vibratório". Se ele estivesse usando o AutoRinger, precisaria apenas configurar a aplicação uma única vez para que todos os dias, no horário das aulas, o aparelho, ao detectar que o usuário se encontra em sala de aula, altere o tipo de toque para "silencioso" automaticamente.

Para funcionar adequadamente, o AutoRinger necessita de várias informações contextuais: dados da agenda de compromissos, localização e preferências do usuário. De modo similar ao ocorrido para o SmartTravel, toda a provisão de contexto é também realizada pelo ConBus, propiciando aos desenvolvedores do AutoRinger uma separação de interesses entre a provisão de informações contextuais e o uso dessas informações pela aplicação propriamente dita. Além das facilidades já discutidas na construção do SmartTravel, deve-se mencionar uma das características mais importantes do ConBus que é explorada no desenvolvimento do AutoRinger, a *reutilização* de módulos de sensoriamento. Neste caso, todos os módulos de sensoriamento necessários (*Date/Time ConUnit, Google Calendar ConUnit, User Preferences ConUnit* e *GPS Receiver ConUnit*) para coletar informações contextuais para o AutoRinger foram reutilizados, a partir dos módulos de sensoriamento desenvolvidos para o SmartTravel.

## 5. Trabalhos Correlatos

Módulos de sensoriamento são elementos comuns a qualquer arquitetura de *middleware* para computação sensível ao contexto. Entretanto, poucos sistemas de *middleware* oferecem abstrações específicas para facilitar o desenvolvimento e integração desses módulos. Essa tarefa envolve problemas como distribuição do sensor (local, remoto ou distribuído), descoberta do sensor (dinâmica ou estática), carregamento e desconexão do módulo de sensoriamento. Context Toolkit [Dey et al. 2001] [Salber et al. 1999], Hydrogen [Hofer et al. 2003] e Contory [Riva 2006] são alguns dos exemplos de sistemas de *middleware* que oferecem abstrações para tratar da integração de módulos de sensoriamento.

No Context Toolkit, os dados de contexto coletados pelos sensores são disponibilizados através dos *context widgets*, que são as abstrações usadas para representar os módulos de sensoriamento neste *framework*. Primeiramente, esses *widgets* devem se registrar em um serviço de registro e descoberta para que, em seguida, um outro componente da arquitetura o descubra e utilize os dados de contexto fornecidos por eles. Portanto, no Context Toolkit, tanto as aplicações quanto os outros componentes da arquitetura devem lidar diretamente com as interfaces desses *widgets*, descobrindo primeiramente onde estão localizados (IP, porta), através dos descobridores (*Discoverers*). O Hydrogen, entretanto, oferece a classe *ContextClient* que fornece o serviço responsável por realizar a implantação dos módulos de sensoriamento ao *framework*, viabilizando a comunicação entre um *adaptador*, que representa um módulo de sensoriamento nessa arquitetura, e o *Servidor de Contexto* da Camada de Gerenciamento. Por outro lado, o Contory utiliza a abstração denominada *Reference Module*, um componente que agrupa diversos sensores com características semelhantes. Porém, esse *middleware* não se preocupa em oferecer facilidades específicas para o desenvolvimento e implantação de módulos de sensoriamento isolados, como acontece com o ConBus, através do *framework ConUnit*. Isto ocorre por-

que seu foco está direcionado à capacidade de integrar múltiplas estratégias de provisão de contexto completas. Por outro lado, através dos componentes *AccessController* e dos *Reference Modules*, o Contory pode, à medida em que for necessário, ativar e desativar fontes de contexto dinamicamente, de modo a manter, em memória, apenas aquelas mais recentemente utilizadas.

No ConBus, todos os componentes são executados integralmente no dispositivo móvel e os *ConUnits* são iniciados apenas quando uma aplicação ou outro componente do *middleware* necessita de suas informações, o que, para dispositivos móveis pode auxiliar no gerenciamento de recursos, como memória, processamento e no consumo de energia. Além disso, uma aplicação não necessita conhecer a interface específica de um módulo de sensoriamento (encapsulado por um *ConUnit*) para obter suas informações, isto é feito através de chamadas à *API das Aplicações*, oferecida pelo ConBus. Deve-se observar, porém, que nem o Context Toolkit, nem o Hydrogen ou o Contory disponibilizam ferramentas especialmente desenvolvidas para facilitar a integração de módulos de sensoriamento junto à plataforma de provisão de contexto. A Tabela 1 resume a comparação feita entre o ConBus e os *frameworks* ou sistemas de *middleware* mencionados nesta Seção, contemplando outras informações sobre as características de cada plataforma em relação à integração (desenvolvimento, reutilização, implantação e gerenciamento do ciclo de vida) de sensores/módulos de sensoriamento.

**Tabela 1. Comparação do ConBus/ConUnit com outros sistemas de middleware**

| | Desenvolvimento | | | Implantação | | | Ciclo de Vida | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Abstração | Reutilização | Modelo de Dados | Mecanismo | Componentizável | Localização sensor | Ativação dinâmica | Instalação dinâmica | Registro/descoberta | Seleção de módulos |
| ConBus | ConUnit | Inst/Cód | par/valor | OSGi | √ | Dep. Módulo | √ | √ | OSGi | Expl/Prim |
| ContextToolkit | Widget | Inst/Cód | par/valor | serviço | √ | Distribuído | — | — | serviço | Expl |
| Contory | Reference | Inst/Cód | par/valor | ? | — | Transp. | × | √ | ? | — |
| Hydrogen | Adaptor | Inst/Cód | O.O. | serviço | √ | Dep. Módulo | × | × | serviço | Expl |

**Legenda:**

| | | | |
|---|---|---|---|
| — | não se aplica ao sistema | Inst | reutilização de instâncias de sensores ou módulos |
| ? | não foi possível identificar. | Transp. | permite qualquer estratégia, deixando transparente às aplicações |
| Cód | reutilização de código de módulos | Expl | indicação explícita do sensor/módulo pela aplicação |
| | | Prim | seleção do primeiro sensor/módulo que atende requisição |

## 6. Conclusões e Trabalhos Futuros

Dada a diversidade de sensores embutidos ou externos ao dispositivos móveis que podem ser usados como fontes de contexto pelas aplicações, a complexidade de desenvolver módulos de sensoriamento de contexto para coletar dados de cada um desses possíveis sensores aumenta significativamente. Muitas vezes, isto dificulta ou até mesmo inibe o desenvolvedor da aplicação em explorar o potencial da sensibilidade ao contexto devido à dificuldade e ao custo envolvidos para obter as informações de contexto necessárias para adaptar o comportamento da sua aplicação. Neste trabalho, foi proposta uma plataforma de *middleware* - ConBus - que trata de pontos fundamentais que podem representar um grande empecilho para o desenvolvimento de aplicações do gênero, tais como: *i)* criação de módulos de sensoriamento com interfaces padronizadas; *ii)* reutilização de módulos de

sensoriamento desenvolvidos por terceiros; *iii)* representação dos dados de contexto independentemente do sensor utilizado; e *iv)* gerenciamento dos módulos de sensoriamento instalados.

Através do ConBus, espera-se que o desenvolvedor possa reutilizar módulos de sensoriamento de contexto existentes e focar mais essencialmente na lógica de negócio da sua aplicação do que na complexidade de captura da informação contextual. Como trabalhos futuros, pretendemos implementar no ConBus algoritmos que, em tempo de execução, façam a seleção do módulo de sensoriamento que melhor atenda às necessidades da aplicação caso haja mais de um para o mesmo tipo de informação de contexto. Há uma série de variáveis e heurísticas de avaliação para cada tipo de contexto e sensor que podem ser consideradas para essa tomada de decisão. Além disto, pretendemos desenvolver protocolos de descoberta dinâmica de módulos de sensoriamento que podem ser compartilhados entre instâncias do ConBus executando em dispositivos diferentes colocalizados. Uma outra possível extensão deste projeto, será o carregamento remoto automático de módulos de sensoriamento usando R-OSGi (Remote OSGi).

## Referências


Alliance., O. (2009). Osgi - the dynamic module system for java. disponível em: http://www.osgi.org, pesquisado em 11/10/2009.

Chen, H. L. (2005). *An intelligent broker architecture for context-aware systems*. Unpublished phd thesis, University of Maryland, Department of Computer Science and Electrical Engineering.

Dey, A., Salber, D., and Abowd, G. (2001). A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction Journal (HCI)*, 16(2-4):97–166.

Henricksen, K. and Indulska, J. (2006). Developing context-aware pervasive computing applications: Models and approach. *Pervasive and Mobile Computing*, 2(1):37–64.

Hofer, T., Schwinger, W., Pichler, M., Leonhartsberger, G., Altmann, J., and Retschitzegger, W. (2003). Context-awareness on mobile devices - the hydrogen approach. In *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on*.

Hong, J. I. and Landay, J. A. (2004). An architecture for privacy-sensitive ubiquitous computing. In *MobiSYS '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 177–189. ACM Press.

Riva, O. (2006). Contory: A middleware for the provisioning of context information on smart phones. In *ACM/IFIP/USENIX 7th International Middleware Conference (Middleware'06)*, Melbourne (Australia).

Salber, D., Dey, A. K., and Abowd, G. D. (1999). The context toolkit: aiding the development of context-enabled applications. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 434–441, New York, NY, USA. ACM.

Weiser, M. (1991). The computer for the twenty-first century. *Scientific American*, pages 94–100.