Uma Arquitetura de Middleware para Provisão de Contexto em Dispositivos Móveis

Marcio Pereira de Sá[1], **Vagner Sacramento**[1], **Ricardo Couto Antunes da Rocha**[1]

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

Abstract. *Even though the great evolution and dissemination of mobile devices and embedded sensors, to develop ubiquitous applications still is a complex task due to the great diversity of sensors, each providing a different data access interface. So, this work proposes a context-aware provisioning architecture that offers a abstraction layer to developers of mobile applications.*

Resumo. *Apesar da grande evolução e disseminação dos dispositivos móveis e sensores acoplados, desenvolver aplicações ubíquas ainda é uma tarefa complexa, principalmente, devido à grande diversidade de sensores, cada qual fornecendo uma interface de acesso diferente. Por isso, este trabalho propõe uma arquitetura de provisão de contexto que ofereça aos desenvolvedores de aplicações móveis uma camada de abstração para padronizar e facilitar o acesso aos dados de contexto fornecidos por diferentes sensores; bem como permitir a reutilização de módulos de sensoriamento desenvolvidos por terceiros.*

1. Introdução

A difusão dos dispositivos móveis tem motivado o desenvolvimento de novas aplicações ubíquas em diferentes domínios. Aplicações financeiras, de entretenimento, educativas, dentre outras, são alguns exemplos mais comuns. Porém, tanto esses quanto outros tipos de aplicações móveis podem ser mais úteis se forem capazes de perceber e reagir a mudanças em seus ambientes físico e/ou computacional em que estão inseridos. Isto pode ser importante devido, principalmente, ao próprio ambiente móvel que, em geral, oferece recursos computacionais limitados (CPU, memória, energia, tamanho e resolução da tela, etc.), bem como uma grande quantidade de variavéis ambientais (localização, luminosidade, temperatura, pessoas co-localizadas, dentre outras) que variam muito ao longo da execução dessas aplicações. Todas essas informações são denominadas informações de contexto. De fato, as aplicações podem fazer uso dessas informações para adaptar seu comportamento de acordo com o contexto corrente, sem a intervenção explícita dos usuários, tornando tais aplicações mais "inteligentes" e eficientes. Desse modo, com a evolução da computação móvel e da computação sensível ao contexto, a computação ubíqua/pervasiva, idealizada por Mark Weiser [Weiser 1991], na década de 80, torna-se, a cada dia, uma realidade para os usuários finais.

É evidente que o emprego da sensibilidade ao contexto faz emergir uma grande quantidade de possibilidades para o desenvolvimento de aplicações móveis, permitindo construir sistemas computacionais mais adaptados ao ambiente móvel e ubíquo. Entretanto, apesar das potenciais possibilidades oferecidas, nota-se que ainda há poucas aplicações sensíveis ao contexto disponíveis. Isso ocorre porque existem ainda muitos

desafios a serem vencidos nesta área do conhecimento. Dentre eles estão a grande quantidade de tecnologias de sensoriamento (por exemplo, vários tipos de receptores GPS ou sensores de temperatura, cada um com uma interface de acesso própria) e a grande diversidade de informações de contexto (localização, luminosidade, dados sobre a agenda de compromisssos dos usuários, uso de CPU e memória, qualidade da rede sem fio, etc.).

Por isso, devido a essa diversidade de informações de contexto e à abundância de tecnologias de sensoriamento, atualmente, um dos principais empecilhos para o desenvolvimento de tais aplicações é a complexidade em desenvolver novos módulos de sensoriamento de contexto (estes módulos são, na verdade, as interfaces de software que permitem às aplicações acessarem os dados coletados pelos sensores) ou reutilizar os existentes desenvolvidos por diferentes grupos de trabalho. Isto acontece porque, em geral, o desenvolvimento desses módulos de sensoriamento demanda muito esforço de programação de baixo nível, pois requer a interação com drivers do sistema ou APIs específicas oferecidas pelos sistemas operacionais ou por terceiros para a coleta de informações, tais como qualidade da rede sem fio, uso de CPU, dados sobre a temperatura ambiente fornecidos por sensores externos, dados das agendas de compromissos disponíveis em *websites*, etc. Problemas como esses ocorrem por causa da falta de padronização das interfaces de acesso ao contexto e da representação das informações de contexto coletadas por sensores desenvolvidos por diferentes grupos de trabalho. Isto dificulta ou, até mesmo, inviabiliza a reutilização de implementações de sensores (módulos de sensoriamento) existentes, pois o desenvolvedor teria que entender a interface de acesso ao contexto de cada sensor, e lidar com codificações e representações externas dos dados, muitas vezes, dependentes de linguagem/plataforma.

Com o propósito de lidar com tais desafios, este trabalho propõe uma arquitetura de *middleware*, denominada *ConBus* (***Con**text **Bus***), que provê serviços de coleta, processamento e disponibilização de informações de contexto para facilitar o desenvolvimento de aplicações móveis.

Este trabalho está organizado da seguinte forma: na Seção 2, são apresentados alguns requisitos desejados por uma arquitetura de provisão de contexto para dispositivos móveis. Na Seção 3, é realizada uma análise sobre alguns trabalhos relacionados à proposta discutida neste artigo. A arquitetura *ConBus* é discutida em detalhes na Seção 4. Finalmente, a Seção 5 discute alguns trabalhos futuros visando à melhoria da arquitetura e, logo em seguida, apresenta as considerações finais sobre o presente trabalho.

2. Requisitos para arquiteturas de provisão de contexto em dispositivos móveis

O uso de informações de contexto em aplicações móveis pode trazer grandes benefícios a seus usuários, visto que permite às aplicações se adaptarem às mudanças do ambiente, utilizando melhor os limitados recursos oferecidos pelos dispositivos móveis. Pelo fato do ambiente móvel ser muito dinâmico, sendo modificado com frequência, há particularidades e características que devem ser levadas em consideração quando se deseja desenvolver aplicações móveis sensíveis ao contexto e, por conseguinte, também devem ser consideradas ao se desenvolver arquiteturas de provisão de contexto para plataformas móveis. Dentre as principais características dos ambientes móveis estão a limitação das capacidades de processamento, armazenamento e memória; limitações físicas dos dispo-

sitivos (tamanho da tela, ausência de teclado físico ou tamanho muito reduzido), pouca duração das baterias, restrições de comunicação, dentre outras. Baseados nas limitações dos dispositivos móveis descritas anteriormente, foram identificados alguns requisitos para se desenvolver arquiteturas de provisão de contexto para aplicações móveis:

- *Menor consumo de recursos*: Toda infra-estrutura de provisão de contexto deve se preocupar em ser o mais eficiente possível, isto é, consumir a menor quantidade de recursos do dispositivo móvel que puder. Pois, se isto não for observado, muitas aplicações móveis podem ficar inviáveis devido ao alto consumo de recursos da própria infra-estrutura que provê contexto a elas.
- *Interface comum e flexível para acesso ao contexto, fornecendo interfaces de comunicação síncrona e assíncrona para acesso à informação de contexto*: Devido às características das aplicações sensíveis ao contexto, que normalmente devem adaptar seus comportamentos de acordo com as mudanças do contexto em que estão inseridas, a interface de comunicação com os sensores disponibilizada deve proporcionar uma forma de comunicação baseada em eventos do tipo *publish/subscribe*, além da comunicação síncrona tradicional, onde é possível simplesmente consultar uma informação específica armazenada pela infra-estrutura (comunicação do tipo requisição/resposta). Portanto, a comunicação assíncrona permite às aplicações se registrarem em alguns tipos de eventos e serem informadas apenas quando estes eventos acontecerem, poupando recursos do sistema.
- *Extensibilidade em relação à integração com novas fontes de contexto:* Ou seja, a arquitetura deve oferecer formas padronizadas, seja através de APIs ou algum outro mecanismo, que facilite a conexão de novos sensores à plataforma de provisão de contexto.
- *Integração com outras plataformas de provisão de contexto:* Em um ambiente ubíquo, pode ser vantajoso para uma plataforma de provisão de contexto se integrar a outras, de forma a utilizar, sempre que possível, os serviços oferecidos por elas, reutilizando módulos de sensoriamento e serviços previamente desenvolvidos e testados.
- *Compartilhamento de informações contextuais:* Esta característica permite a troca de informações entre diferentes dispositivos. Por exemplo, se um dispositivo não dispõe de um sensor de localização e está fisicamente próximo a um outro dispositivo que possui um receptor GPS embutido, se eles puderem se comunicar, o dispositivo que não dispõe de sensor de localização poderá utilizar as informações de localização fornecidas pelo receptor GPS do outro dispositivo para inferir sua localização corrente.
- *Independência de plataforma:* Uma arquitetura de provisão de contexto para aplicações móveis deveria oferecer seus serviços independentemente da plataforma utilizada (Windows Mobile, Symbian, Android, etc.). Isto facilitaria a reutilização de módulos de sensoriamento e serviços oferecidos pela arquitetura.

Além desses requisitos, o uso de um repositório na Web para armazenamento de módulos de sensoriamento seria muito importante, pois auxiliaria na descoberta e reutilização de módulos desenvolvidos por outros programadores, evitando recriar módulos, testá-los e validá-los desnecessariamente.

3. Trabalhos Correlatos

Apesar dos módulos de sensoriamento serem elementos comuns a qualquer arquitetura de *middleware* para computação sensível ao contexto, poucos sistemas de *middleware* oferecem abstrações específicas para facilitar o desenvolvimento e integração desses módulos. Esta tarefa envolve problemas como distribuição do sensor (local, remoto ou distribuído), descoberta do sensor (dinâmica ou estática), carregamento e desconexão do módulo de sensoriamento. Dentre as plataformas de *middleware* que oferecem abstrações para tratar a integração de módulos de sensoriamento, destacam-se: Context Toolkit [Dey et al. 2001], CFN/Solar [Chen et al. 2004] e Contory [Riva 2006].

O Context Toolkit [Dey et al. 2001] propõe o conceito de *Context Widget*: um componente de software que implementa a abstração de acesso a um módulo de sensoriamento, responsável por prover informações contextuais sobre uma entidade particular. Context Toolkit oferece ainda abstrações que permitem implementar a composição de *widgets*, que representem a agregação de informações contextuais. CFN/Solar [Chen et al. 2004] provê um mecanismo distribuído de composição de sensores, formando uma rede de sensores distribuídos que agreguem informações contextuais e as distribuam eficientemente para aplicações executando em diferentes pontos de uma rede. Dois sistemas de middleware, em particular, provêem acesso eficiente de módulos de sensoriamento em dispositivos móveis: Hydrogen e Contory. Contory [Riva 2006] provê às aplicações transparência de distribuição do mecanismo de provisão de contexto, podendo o módulo de sensoriamento ser local, remoto ou *ad hoc* (em outro dispositivo móvel). Hydrogen [Hofer et al. 2003], por outro lado, não oferece abstrações específicas que facilitem o desenvolvimento de módulos de sensoriamento, cabendo ao desenvolvedor a integração do módulo nos mecanismos do middleware.

A preocupação fundamental desses e outros sistemas é facilitar o desenvolvimento das aplicações e o acesso aos módulos de sensoriamento. Tipicamente, eles levam em consideração que tais módulos já estão pré-carregados ou que outro desenvolvedor já se encarregou de prover os módulos e adequá-los ao middleware. O desenvolvimento e a implantação de módulos de sensoriamento requerem, entretanto, mecanismos e abstrações específicos que facilitem a sua integração com o middleware de computação sensível ao contexto e o seu posterior reconhecimento pelas aplicações. A complexidade no desenvolvimento desses módulos é ainda maior quando eles provêem contextos lógicos, resultado da composição de outros módulos de sensoriamento. Além disso, uma solução de middleware precisa ainda ser adequada às limitações dos dispositivos móveis no qual as aplicações estarão executando.

4. Arquitetura ConBus

O projeto *ConBus* foi desenvolvido visando oferecer uma camada de abstração que facilite o desenvolvimento de aplicações sensíveis ao contexto em dispositivos móveis, tais como PDAs e *SmartPhones*. De fato, na arquitetura *ConBus*, sempre que uma fonte de contexto (sensor) tiver que ser substituída por outra equivalente ou tiver que ser modificada, a aplicação que utiliza os dados de contexto fornecidos por ela não precisará ser modificada também. Isto facilita a manutenção de aplicações sensíveis ao contexto, na medida em que isola das aplicações as modificações feitas nos módulos de coleta, processamento, armazenamento e disponibilização de contexto.

A Figura 1 ilustra a arquitetura *ConBus*. Pode-se notar, nesta figura, que esta arquitetura é constituída por três camadas: *Camada de Sensores*, *Framework ConBus* e *Camada de Aplicação*. De acordo com a Engenharia de Software, uma arquitetura em camadas permite aplicar, também ao projeto de sistemas sensíveis ao contexto, um importante princípio arquitetural: a separação de interesses que, neste caso, significa desacoplar os mecanismos de coleta e processamento de dados contextuais da provisão de informações de contexto para a aplicação. Isto permite aos desenvolvedores das aplicações móveis se preocuparem, essencialmente, com questões relativas à lógica da aplicação, podendo assim aumentar a sua produtividade e melhorar a utilidade da aplicação em questão. Cada camada da arquitetura *ConBus* é discutida em maiores detalhes nas subseções seguintes.Por outro lado, pode-se indagar se o uso de simples bibliotecas de provisão de contexto fornecidas pelos sistemas operacionais ou desenvolvidas por terceiros não seria suficiente para se desenvolver a grande maioria das novas aplicações sensíveis ao contexto. Deve-se observar, entretanto, a limitação de tais bibliotecas, visto que elas, normalmente, não fornecem serviços mais complexos e que se tornaram muito importantes para as aplicações sensíveis ao contexto atuais, como comunicação assíncrona entre os sensores e as aplicações, interpretadores de contexto que podem ser reutilizados por mais de uma aplicação, além da falta de padronização tanto na representação dos dados contextuais quanto nas interfaces para acesso a esses mesmos dados por parte das aplicações.

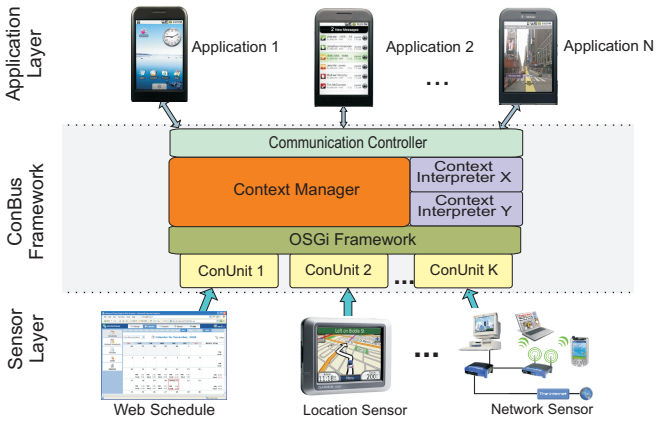

Figure 1. Arquitetura ConBus - Visão Geral.

4.1. Camada de Sensores

A primeira camada a ser discutida é a Camada de Sensores (*Sensor Layer*) ilustrada na Figura 1. Ela é constituída por todas as fontes de contexto (sensores) que fornecem informações ao *framework ConBus* e, por conseguinte, a todas as aplicações sensíveis ao contexto. É importante observar que, neste trabalho, o termo *sensor* é empregado com um significado mais amplo, indicando qualquer dispositivo físico (hardware) ou lógico (software) capaz de coletar alguma informação contextual relevante para melhorar a execução de aplicações, bem como facilitar a interação dessas aplicações com seus usuários. Desse modo, são considerados sensores os dispositivos que coletam dados do ambiente (temperatura, pressão, umidade, luminosidade, etc), do usuário (agenda de compromissos, preferências, pessoas co-localizadas, dentre outras) e, ainda, dados relativos ao próprio ambiente computacional (nível de bateria, uso de CPU e memória, qualidade da rede

sem fio, tamanho, orientação e resolução da tela, e muitas outras.). Por outro lado, o termo *módulo de sensoriamento* de um determinado sensor, indica a interface de software que permite às aplicações ou a qualquer outro software obter as informações contextuais coletadas pelo sensor correspondente. Os módulos de sensoriamento são, portanto, os intermediadores entre os sensores propriamente ditos e as aplicações que utilizam seus dados.

4.2. Framework ConBus

O *Framework ConBus* é a camada intermediária que gerencia todas as informações contextuais coletadas pelas fontes de contexto da Camada de Sensores e que, após serem processadas adequadamente pelo próprio *framework ConBus*, são fornecidas a todas as aplicações sensíveis ao contexto que fazem uso desses dados. Esta camada é composta pelos componentes descritos a seguir. O Gerenciador de Contexto (*Context Manager*), responsável pelo controle de todos os recursos oferecidos pelo *framework*, ou seja, o controle de todas as informações coletadas pelos sensores acoplados ao *ConBus*, bem como a responsabilidade de disponibilizá-las adequadamente às aplicações sensíveis ao contexto. Além do Gerenciador de Contexto, há também os *ConUnits* (Unidades de Contexto) que interligam, de forma padronizada, as fontes de contexto ao *ConBus* (veja a Subseção 4.4). Há também o *Communicator Controller* cuja função principal é permitir a comunicação entre o *framework ConBus* e as aplicações sensíveis ao contexto. Outros componentes importantes são os Interpretadores de Contexto (*Context Interpreters*), que realizam tarefas importantes para a arquitetura, inferindo novas informações de contexto a partir de dados de contexto de mais baixo nível. Por exemplo, pode-se desenvolver um interpretador específico que, a partir das coordenadas geográficas (latitude e longitude) coletadas por um receptor GPS, pode inferir se um determinado usuário ou dispositivo encontra-se ou não em uma dada sala de um edifício.

É importante notar que, quando se permite que se possa utilizar módulos interpretadores instalados dentro do *framework ConBus* (apesar de ser mais natural que eles façam parte das aplicações, por serem, muitas vezes, específicos demais), permite-se que módulos desenvolvidos por terceiros, a exemplo dos *ConUnits*, possam também ser reutilizados por outros desenvolvedores de aplicações que necessitam de serviços de interpretação semelhantes. Além dos componentes citados anteriormente, há ainda o *framework OSGi* [Alliance. 2009] cuja função é permitir o acoplamento de novos *ConUnits* e *Interpretadores de Contexto* ao *Context Manager* (na terminologia *OSGi*, cada componente/módulo acoplado ao *framework OSGi* é denominado *bundle*). O *OSGi* permite ainda que *ConUnits* ou *Interpretadores de Contexto* sejam instalados, desistalados, iniciados, desligados, e até, atualizados dinamicamente sem a necessidade de reiniciar a execucão de todo o *framework ConBus*. Isto permite que o próprio *ConBus* seja sensível ao contexto e, sempre que necessário, possa instalar ou desistalar, iniciar ou desligar *ConUnits* ou Interpretadores de Contexto, visando poupar recursos computacionais, como, energia, CPU e memória, quando estes estiverem escassos. Além disso, pode-se realizar, remotamente e de forma automatizada, atualizações em *ConUnits* ou Interpretadores de Contexto.

4.3. Camada de Aplicação

Todas as aplicações sensíveis ao contexto que utilizam as informações contextuais fornecidas pelo *framework ConBus* fazem parte da Camada de Aplicação (*Application*

Layer). Para usufruir das informações de contexto providas pelo *ConBus*, uma aplicação necessita apenas conhecer a URL (neste caso, http://localhost:8585) da instância do *framework ConBus* que está sendo executada no dispositivo móvel correspondente e, em seguida, invocar os métodos desejados com os devidos parâmetros oferecidos pela interface de acesso ao contexto, usando, para isso, interfaces síncronas e assíncronas providas pela API das Aplicações (*Application API*). Esta API define um *Web Service* local baseado em REST [O'Reilly. 2009], através do qual as aplicações podem enviar requisições para a instância do *ConBus* que se encontra em execução.

4.4. ConUnit - Arquitetura e Implementação

Pelo fato de ser o módulo responsável pela interação com os sensores (fontes de contexto), os *ConUnits*, cuja arquitetura é ilustrada na Figura 2, são o elo entre estas fontes de contexto e o *ConBus*. Um *ConUnit* é constituído por duas partes principais: o *Sensor Communicator* e o *ConBus Communicator*.

O *Sensor Communicator* é a parte dependente de sensor e, por isso, o programador do *ConUnit* correspondente deverá implementar uma interface Java denominada Interface de Conversão de Dados de Contexto (*Context Data Conversion Interface*); é através da implementação dessa interface que será realizada a conversão dos dados específicos de cada sensor para uma forma padronizada. O *ConBus Communicator* é a parte independente de sensor; é constituído pelos seguintes componentes: o *Synchronous Controller* é responsável pela comunicação síncrona entre o *ConUnit* e o *Context Manager*; de forma análoga, o *Asynchronous Controller*, viabiliza a comunicação assíncrona, baseada em eventos, entre o *ConUnit* e o *Context Manager*; há ainda, o *ConUnit Manager* que controla todas as funções do *ConUnit*, tais como, o acesso aos dados coletados pelos sensores, seu adequado tratamento e o posterior envio desses dados, já padronizados, ao *Context Manager*.

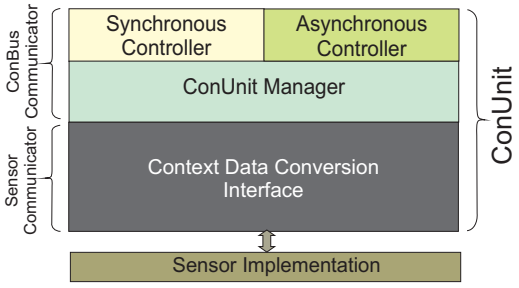

Figure 2. ConUnit - Estrutura Interna.

Para mostrar a viabilidade da arquitetura *ConBus*, uma implementação protótipo está sendo desenvolvida para o sistema operacional Android 1.5, usando a linguagem Java 6 e a implementação Apache Felix 1.4 do framework OSGi. Até o presente momento, já se encontra implementado o *ConUnit framework* cuja principal função é facilitar o desenvolvimento de *ConUnits*. Além desse *framework*, há uma implementação básica do *Context Manager* apta a receber apenas requisições síncronas das aplicações.

5. Trabalhos Futuros e Considerações Finais

Foi proposto, neste trabalho, a arquitetura *ConBus*, com o objetivo de facilitar o desenvolvimento de aplicações móveis sensíveis ao contexto. Esta oferece uma camada

de abstração para o desenvolvimento de aplicações móveis sensíveis ao contexto 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 padronizados; ii) reutilização de módulos de sensoriamento desenvolvidos por terceiros e iii) representação dos dados de contexto independentemente do sensor utilizado.

Como próximos passos, pretende-se desenvolver aplicações sensíveis ao contexto, como estudos de caso, que utilizem os serviços e as informações disponibilizadas pelo *ConBus*. Além disto, devem ser implementados, tanto a comunicação assíncrona entre o *Context Manager* e as aplicações sensíveis ao contexto, quanto alguns interpretadores de contexto específicos para auxiliar as aplicações em suas atividades de interpretação de dados contextuais. Pretende-se ainda integrar o *ConBus*, como uma possível fonte de contexto, a outras arquiteturas de provisão de contexto de uso mais geral, tais como MoCA [Sacramento et al. 2004], ECORA [Padovitz et al. 2008] e SOCAM [Gu et al. 2005].

References

Alliance., O. (2009). Osgi - the dynamic module system for java. disponível em: http://www.osgi.org/main/homepage, pesquisado em 19/03/2009.

Chen, G., Li, M., and Kotz, D. (2004). Design and implementation of a large-scale context fusion network. In *Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004. The First Annual International Conference on*, pages 246–255.

Dey, A., Salber, D., and Abowd, G. (2001). A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications.

Gu, T., Pung, H. K., and Zhang, D. Q. (2005). A service-oriented middleware for building context-aware services. *Journal of Network and Computer Applications*, 28(1):1–18.

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*, pages 10 pp.+.

O'Reilly. (2009). Implementing rest web services: Best practices and guidelines. disponível em: http://www.xfront.com/rest-web-services.html, pesquisado em 19/03/2009.

Padovitz, A., Loke, S. W., and Zaslavsky, A. (2008). The ecora framework: A hybrid architecture for context-oriented pervasive computing. *Pervasive Mob. Comput.*, 4(2):182–215.

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).

Sacramento, V., Endler, M., Rubinsztejn, H. K., Lima, L. S., Goncalves, K., Nascimento, F. N., and Bueno, G. A. (2004). Moca: A middleware for developing collaborative applications for mobile users. *IEEE Distributed Systems Online*, 5(10):2.

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