Um *Framework* de Desenvolvimento de Aplicações Ubíquas em Ambientes Inteligentes

Matheus Erthal[1], **Douglas Mareli**[1], **David Barreto**[1], **Orlando Loques**[1]

[1]Instituto de Computação – Universidade Federal Fluminense (UFF)
Niterói – RJ – Brazil

{merthal,dmareli,dbarreto,loques}@ic.uff.br

Resumo.

1. Introdução

A Computação Ubíqua, como proposta por Weiser na década de 90 [Weiser 1991], prevê uma mudança no paradigma de interação entre o usuário e os sistemas computacionais. Weiser previu o surgimento do que chamou de ”computação calma”, onde a interação entre os usuários e os computadores ocorre de forma indireta. Uma aplicação ubíqua identifica as necessidades do usuário obtendo informação de contexto através de sensores, e provê serviços através de atuadores. Este tipo de sistema de aplicações está geralmente associado a um espaço denominado de ambiente inteligente [Augusto and McCullagh 2007].

A construção e manipulação de aplicações ubíquas representam um grande desafio para desenvolvedores com pouco conhecimento técnico e recursos escassos. Alguns problemas estão mais em evidência como a diversidade de requisitos não funcionais característicos de sistemas distribuídos, como segurança e tolerância a falhas. Para construção e teste de aplicações há a necessidade de um contingente de recursos como dispositivos embarcados e espaço físico. Há uma dificuldade de estabelecer um protocolo comum de comunicação em boa parte destes dispositivos. E por fim, a quantidade e variedade de informações de contexto disponível no ambiente dificulta a interatividade das aplicações ubíquas. Atendendo a esta demanda é proposto um *framework* com o objetivo de facilitar a aplicação dos conceitos de computação ubíqua em ambientes inteligentes de forma simples e confiável.

Muitos trabalhos como [Helal et al. 2005, Cardoso and Sztajnberg 2006, Ranganathan et al. 2005] tentam atingir esse objetivo.
Em [Augusto and McCullagh 2007] são apontados desafios na aquisição de conhecimentos do ambiente. Em [Helal et al. 2005], é proposto um *middleware* entre a camada física, a qual compreende os sensores e atuadores, e a camada de aplicação, na qual se encontram o ambiente de desenvolvimento e as aplicações. Em [Cardoso and Sztajnberg 2006] são propostos serviços para gerenciar, no nível de *middleware*, componentes representativos do ambiente. Em [Ranganathan et al. 2005], sabendo-se que um ambiente inteligente pode possuir uma variedade imensa de dispositivos, propõe se uma estrutura de representação dos componentes da camada física através de ontologia. Esta estruturação permite ampliar o escopo de operações de suporte sobre um ambiente inteligente.

Neste trabalho é proposto um *framework* de desenvolvimento de aplicações ubíquas em ambientes inteligentes. O objetivo é dar suporte a programação, teste e

execução de aplicações para ambientes inteligentes permitindo lidar de forma consistente com sistemas de grande complexidade. O *framework* destaca-se por cobrir grande parte dos desafios destacados na computação ubíqua [de Araujo 2003] como tratamento da heterogeneidade de dispositivos, tratamento de informações de contexto e descoberta de serviços. A heterogeneidade é tratada através da definição de um modelo de componentes distribuídos, no qual o componente básico tem uma estrutura uniforme definida como um agente de recurso. Segundo [Xavier 2006], agente de recurso é a entidade de coleta de informações contexto. Neste trabalho, esta definição é ampliada para qualquer módulo de interação com elementos ambiente inteligente. Para o tratamento de informações de contexto é proposto um modelo de regras que consiste de um conjunto de interpretadores de contexto. Além disso, no *framework* é proposto uma interface de prototipagem [**?**] que permite a visualização e o teste de aplicações ubíquas mesclando componentes reais e virtuais.

A qualidade do suporte do *framework* foi avaliada durante o processo de transformação de uma aplicação com funcionamento estritamente local em uma aplicação ubíqua. Com isso foi possível concluir provar conceitualmente que o *framework* facilita o processo de construção de aplicações ubíquas. Para testar a eficiência do modelo de regras foi construída uma aplicação que explora os mecanismos de comunicação utilizados no modelo de componentes distribuídos e as informações básicas de um agente de recurso como localização e identificação de tipo.

Este artigo é organizado da seguinte forma: a Seção 2 apresenta os conceitos básicos utilizados ao longo do texto. A Seção 3 apresenta a arquitetura geral do SmartAndroid incluindo o modelo de componentes distribuídos e de regras. Na Seção 4 é apresentada uma prova de conceito demonstrando a eficiência do *framework* em construir aplicações ubíquas. Após a avaliação, a Seção 5 apresenta uma comparação com trabalhos relacionados. E para finalizar, a Seção 7 apresenta as conclusões e trabalhos futuros.

2. Conceitos Básicos

2.1. Computação Ciente de Contexto

2.2. Prototipagem de Aplicações Pervasivas

Figura 1

3. Proposta do Framework

Figura 2

O framework provê facilidades e padrões de programação a aplicações cientes de contexto focadas em ambientes inteligentes (*smarthomes*).

Ao invés de se lidar diretamente com sensores e atuadores espalhados no ambiente, o *framework* utiliza, como abstração, componentes chamados Agentes de Recurso (AR). Os ARs são elementos de primeira ordem da plataforma, representando recursos de hardware ou software disponíveis no ambiente. Eles têm como responsabilidade coletar as diversas medidas dos sensores ou componentes de *software* presentes no contexto de execução do das aplicações [**?**]. Os ARs resolvem a complexidade de tratamento com os componente de baixo nível e disponibilizam interfaces padrões de interação tanto local

Figura 1. Interface de Prototipagem

como em rede. Em ambientes inteligentes podem ser criados ARs para representar, por exemplo: uma televisão que expõe suas informações de contexto e provê serviços para as aplicações; sensores de temperatura da casa; sensores de vazamento de gás na cozinha; um medidor de pressão arterial; um atuador para fechar as janelas; etc.

O AR é a unidade básica de modularização do ambiente distribuído proposto pelo *framework*, assim sendo, o mesmo pode também ser utilizado na modelagem das aplicações. Se uma aplicação necessita de uma entidade distribuída, pode representá-la através de um AR. Por exemplo, um desenvolvedor poderia construir um jogo, representando através de ARs os jogadores, o tabuleiro, os monstros, recursos do cenário, etc.

Para possibilitar a comunicação entre os diferentes ARs, o SmartAndroid dispõe de um serviço que gerencia estes recursos, e mantém informações de identificação e localização dos mesmos, dentre outras. Este serviço chama-se Serviço de Gerenciamento de Agentes de Recurso (SGAR) e é instanciado apenas uma vez no sistema distribuído. Como apresentado na Figura **??**, o SGAR é composto por três componentes, ou serviços: o Serviço de Registro de Recurso (SRR), o Serviço de Descoberta de Recursos (SDR) e o Serviço de Localização de Recursos (SLR). A diferença entre o SDR e o SLR está no fato de que o primeiro se refere à localização na rede de computadores, e o segundo se refere à localização física do dispositivo no ambiente.

O SGAR tem por objetivo aumentar a visibilidade das informações de contexto dos ARs e tornar a comunicação transparente para o programador das aplicações.

3.1. Modelo de Componentes Distribuídos

3.2. Comunicação

Figura: Comunicação direta e indireta em alto nível

A principal forma de comunicação no SmartAndroid é através de um mecanismo de publica-subscreve (*publish-subscribe*), também chamado de comunicação por eventos. Este paradigma corresponde à uma comunicação assíncrona, que envolve o registro de interesse por parte da entidade interessada na entidade de interesse. As entidades são

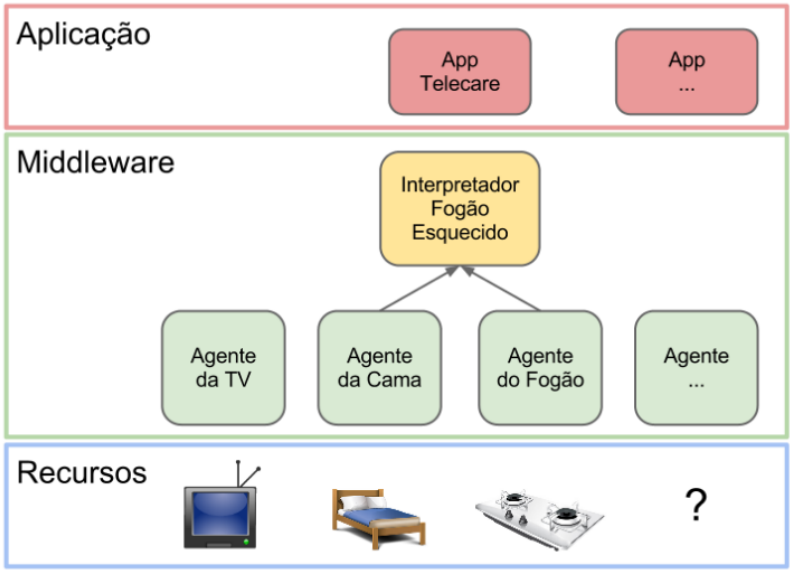

Figura 2. Arquitetura do *Framework*

mapeadas no SmartAndroid em ARs, e uma entidade pode registrar seu interesse no contexto de uma outra qualquer, co-localizada ou remota. Conforme o contexto da entidade de interesse varia no tempo, esta notifica aos interessados, que desempenham suas ações relacionadas.

3.3. Suporte ao Gerenciamento de Recursos

Figura 4

Figura 5

Figura 6

3.4. Modelo de Contexto

Uma infraestrutura para construção de aplicações cientes de contexto (ou aplicações ubíquas) pode ter uma abordagem centralizada ou distribuída. Em abordagens centralizadas se faz necessário a utilização de um serviço de gerenciamento de contexto, responsável por manter a informação contextual e oferecer interfaces para subscrições e consultas. Este tipo de abordagem corresponde à uma arquitetura do tipo *blackboard* (ou quadro-negro), onde uma entidade envia uma mensagem para uma memória compartilhado comum, e também pode se subscrever para receber mensagens que respeitam algum padrão especificado [Winograd 2001]. Todas as comunicações ocorrem através de um servidor centralizado e as mensagens são redirecionadas para os interessados.

Este trabalho adotou uma abordagem distribuída, onde a informação de contexto não é armazenada em um servidor centralizado, mas é mantida pelos ARs. O processo de aquisição do contexto envolve a descoberta do AR de interesse através do SDR, com a posterior subscrição do mesmo, como apresentado na Subsessão 3.2.

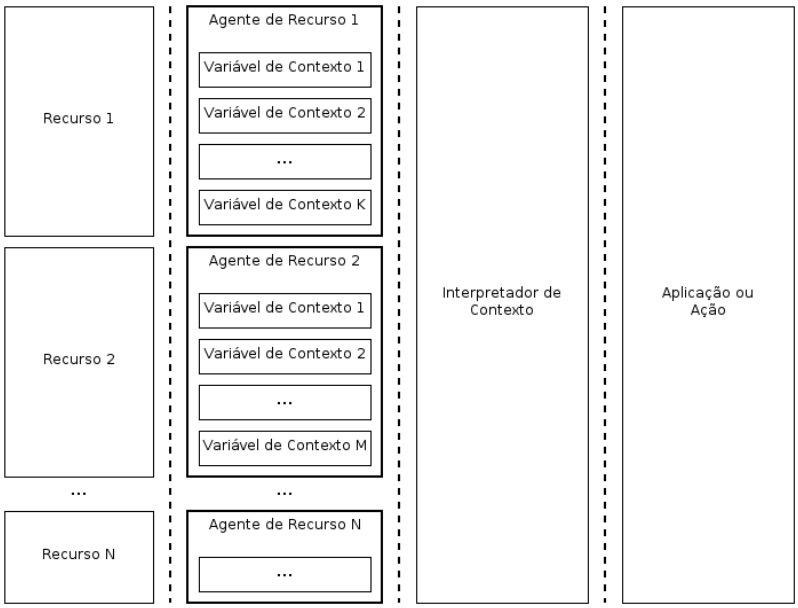

Figura 3. Camada de Interpretação de Regra

3.4.1. Variáveis de Contexto e Operações

As informações de contexto respectivas de cada AR são expostas através de Variáveis de Contexto (VC). Se, por exemplo, há um agente para a televisão registrado no sistema, possivelmente ele provê VCs para qual a programação que está sendo exibida, qual a programação agendada, se a própria televisão está ligada, se está gravando alguma programação, enfim, tudo que diz respeito ao estado da televisão e é efetivamente coletado.

As VCs possibilitam que as informações de contexto sejam acessadas sem que haja um componente centralizado armazenando-as e disponibilizando-as,

3.4.2. Interpretador de Contexto

Figura: Figura que permita mostrar o funcionamento passo-a-passo do IC

4. Avaliação

Figura

5. Trabalhos Relacionados

6. Conclusão e Trabalhos Futuros

Referências

Abowd, G., Dey, A., Brown, P., Davies, N., Smith, M., and Steggles, P. (1999). Towards a better understanding of context and context-awareness. In *Handheld and Ubiquitous Computing*, pages 304–307. Springer.

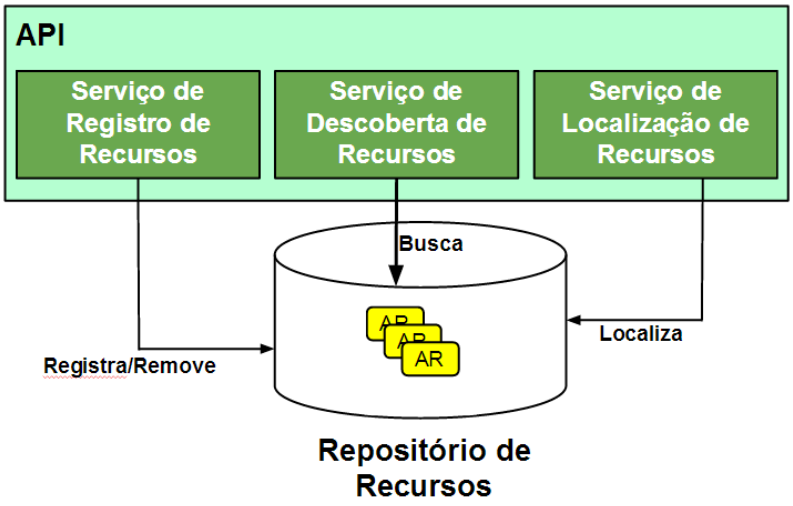

Figura 4. Serviços de Suporte

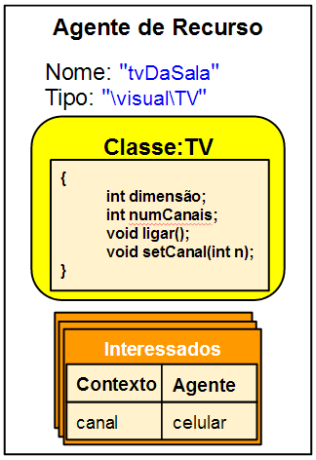

Figura 5. Instância do Agente de Recurso

Augusto, J. and McCullagh, P. (2007). Ambient intelligence: Concepts and applications. *Computer Science and Information Systems/ComSIS*, 4(1):1–26.

Cardoso, L. and Sztajnberg, A. (2006). Self-adaptive applications using ADL contracts. *Self-Managed Networks, Systems,*, pages 87–101.

Chen, G. and Kotz, D. (2002). Solar : An Open Platform for Context-Aware Mobile Applications. (June):41–47.

Chen, Y.S. and Chen, I.C. and Chang, W. (2010). Context-aware services based on OSGi for smart homes. *Ubi-media Computing (U-Media), 2010 3rd IEEE International Conference on*, 11:392.

de Araujo, R. (2003). Computação ubíqua: Princípios, tecnologias e desafios. *XXI Simpósio Brasileiro de Redes de . . .*, pages 45–115.

Dey, A., Abowd, G., and Salber, D. (2001). A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. *Human-Computer Interaction*, 16(2):97–166.

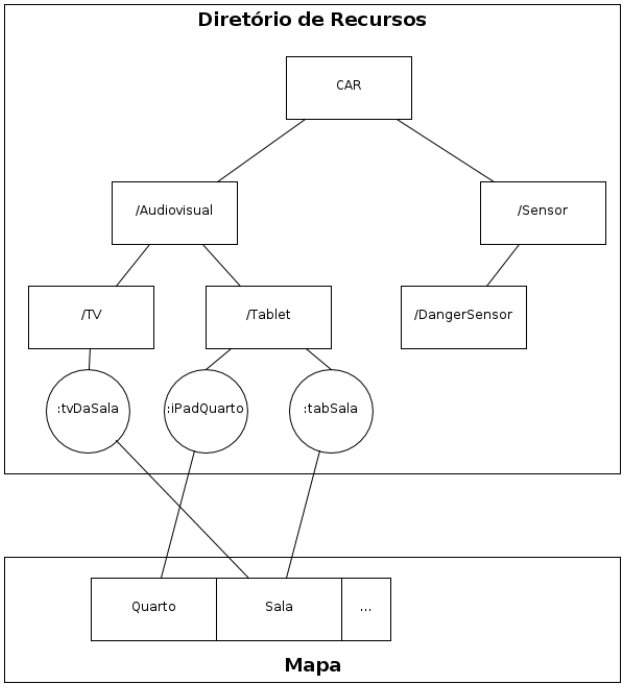

Figura 6. Repositório de Recursos

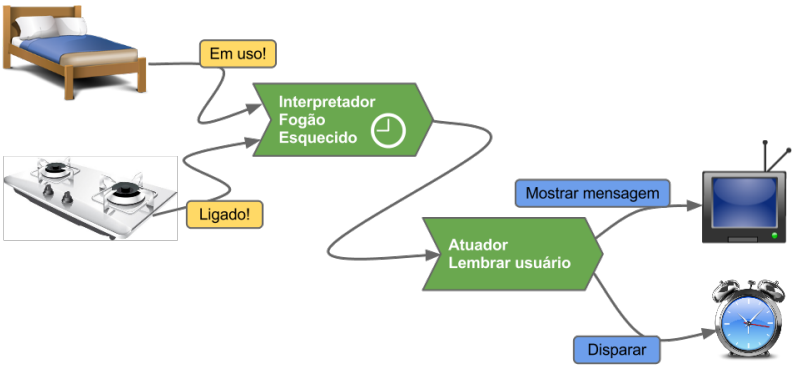

Figura 7. Interpretador de Regra

Helal, S., Mann, W., El-Zabadani, H., King, J., Kaddoura, Y., and Jansen, E. (2005). The Gator Tech Smart House: a programmable pervasive space. *Computer*, 38(3):50–60.

Lee, Y., Iyengar, S., Min, C., Ju, Y., Kang, S., Park, T., Lee, J., Rhee, Y., and Song, J. (2012). Mobicon: a mobile context-monitoring platform. *Communications of the ACM*, 55(3):54–65.

Liu, H. and Parashar, M. (2003). Dios++: A framework for rule-based autonomic management of distributed scientific applications. *Euro-Par 2003 Parallel Processing*, pages 66–73.

Ranganathan, A., Chetan, S., Al-Muhtadi, J., Campbell, R., and Mickunas, M. (2005). Olympus: A high-level programming model for pervasive computing environments. In *Pervasive Computing and Communications, 2005. PerCom 2005. Third IEEE International Conference on*, pages 7–16. IEEE.

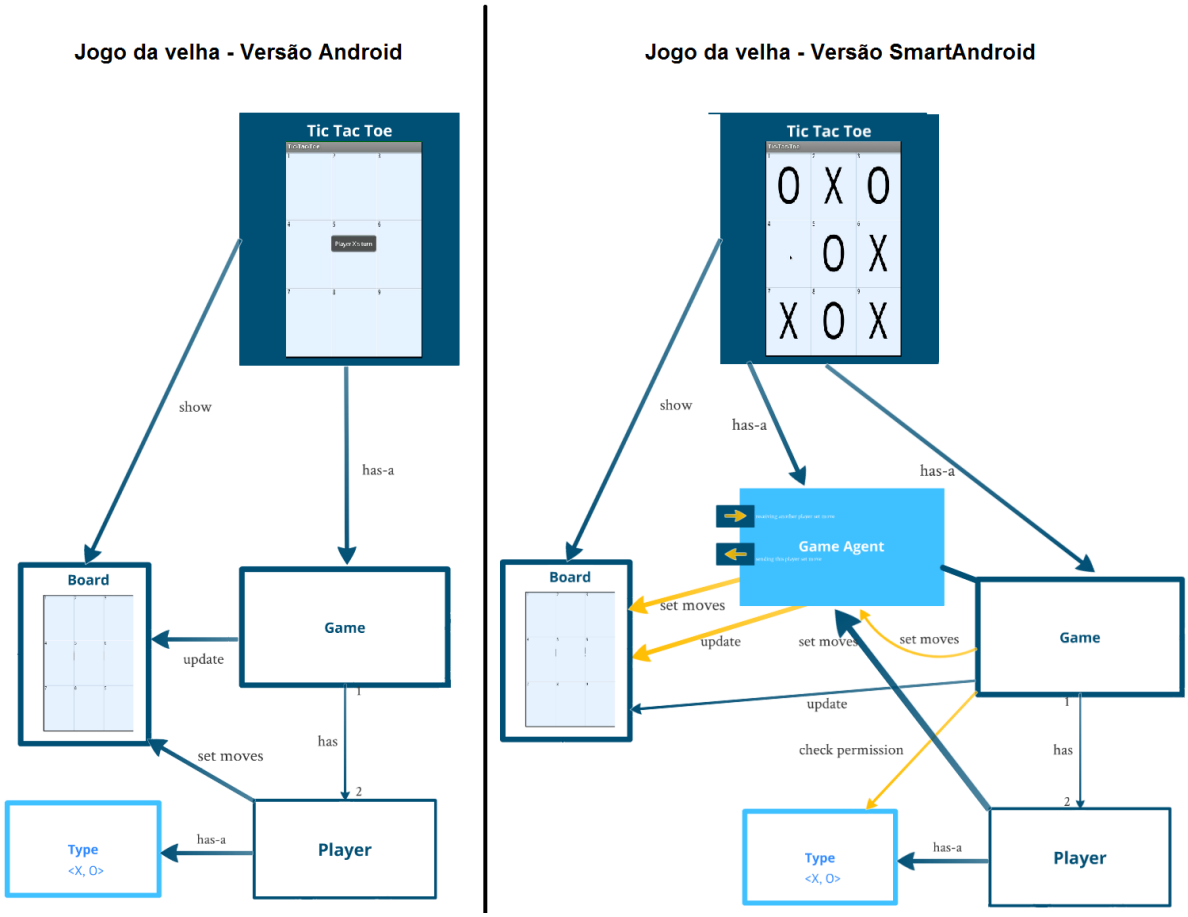

Figura 8. Comparação entre os modelos das aplicações Android e SmartAndroid

Sudha, R., Rajagopalan, M., Selvanayaki, M., and Selvi, S. (2007). Ubiquitous semantic space: A context-aware and coordination middleware for ubiquitous computing. In *Communication Systems Software and Middleware, 2007. COMSWARE 2007. 2nd International Conference on*, pages 1–7. IEEE.

Wang, Q. (2005). Towards a rule model for self-adaptive software. *ACM SIGSOFT Software Engineering Notes*, 30(1):8.

Weis, T., Knoll, M., and Ulbrich, A. (2007). Rapid prototyping for pervasive applications. *IEEE Pervasive*.

Weiser, M. (1991). The computer for the 21st century. *Scientific American*, 265(3):94–104.

Winograd, T. (2001). Architectures for context. *Human-Computer Interaction*, 16(2):401–419.