Interpretação de Contexto 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 usuário e os computadores ocorre de forma indireta. O sistema identifica as necessidades do usuário obtendo informação de contexto através de sensores, e provê serviços através de atuadores.

A Computação Ubíqua está intimamente relacionada com uma área de particular interesse para este trabalho, chamada de Computação Ciente de Contexto [Dey et al. 2001], também chamada de Computação Sensível ao Contexto. Área esta que vem crescendo devido à riqueza de informação que só o contexto pode dar, ao invés de se esperar que o usuário entre com todas as informações úteis no sistema. Um exemplo deste tipo de sistema bem conhecido é o site tocador de músicas Last.fm, que constrói um perfil do usuário baseado no que ele tem buscado, no seu histórico, nas músicas que tem pulado, nas músicas que disse que não gostou, etc, em seguida o sistema provê um serviço de maior qualidade. Outro exemplo é o sistema de propagandas Adwords do Google, que sugere propagandas baseado nas buscas do usuário, oferecendo assim propagandas às quais o usuário pode se realmente interessar.

A informação de contexto está presente não apenas no nível de software, mas também no nível de dispositivos físicos. Na Computação Ubíqua o contexto é qualquer coisa fisicamente mensurável ou detectável, como a presença da pessoa, a hora do dia ou condições atmosféricas [Coulouris et al. 2005]. Com os avanços recentes nas tecnologias de comunicação sem fio, assim como nos dispositivos móveis, abre-se espaço para o, já previsto, crescimento da Computação Ubíqua [Coulouris et al. 2005, Lyytinen and Yoo 2002], também chamada de Computação Pervasiva [Saha and Mukherjee 2003, Satyanarayanan 2001], Inteligência Ambiente (AmI) [Augusto and McCullagh 2007], ou outros [Ranganathan et al. 2005, Augusto and McCullagh 2007].

Pode-se ver como a Computação Ubíqua vem se popularizando já nos dias de hoje. Carros novos contém diversos sensores e atuadores que prestam serviços ao motorista e passageiros, sem que se deem conta; sensores de presença acionam lâmpadas nos corredores à noite; pedágios identificam a presença de veículos que pagam pelo serviço mensalmente, e abrem a cancela automaticamente; e outros. Entretanto, percebe-se que todos estes serviços funcionam em um nível menor: eles não compartilham informações de contexto entre si e nem disponibilizam suas funcionalidades para outros.

Para suprir a necessidade de maior interação entre os recursos distribuídos no ambiente, este trabalho propõe um *framework*, que inclui suporte conceitual e de infraestrutura, para a construção de aplicações ubíquas. Uma camada intermediária se dispõe, entre os recursos e as aplicações, possibilitando uma fácil manipulação do contexto. Do ponto de vista no desenvolvimento de aplicações, esta abstração provê uma separação de interesses, permitindo que os desenvolvedores se preocupem com a lógica das aplicações, e não com os detalhes de comunicação dos recursos. Tomando o compartilhamento de informações como premissa básica, é fácil imaginar diversas aplicações rodando sobre o mesmo ambiente, como por exemplo: aplicações de vigilância da casa, aplicações de cuidados domiciliares (*homecare*), sistemas de combate a incêndio (capazes inclusive de destrancar portas, por exemplo), dentre outras. É fato que este tipo de sistema traz uma série de questões de segurança, mas estas estão também sendo estudadas.

No modelo adotado, cada recurso é representado no sistema por um Agente de Recurso (AR), que por sua vez expõe suas informações de contexto através de Variáveis de Contexto (VC). Por exemplo, o AR da TV poderia expor as VCs: "está ligada", "localização", "canal", "volume", etc; o AR do ar-condicionado poderia expor a "temperatura do cômodo", seu "modo de operação"; a lâmpada poderia dizer se "está ligada"; o acelerômetro do celular poderia identificar se a pessoas "está sentada", "está andando", etc; e muitos outro recursos poderiam ser também incorporados ao sistema. Os ARs são entidades distribuídas mas que, no entanto, podem ser encontradas através do Serviço de Descoberta de Recursos (SDR). A comunicação com os mesmos ocorre através de um mecanismo de publica-subscreve (*publish-subscribe*), onde uns ARs podem registrar seu interesse nas VCs de outros ARs, sendo posteriormente notificados quando da modificação dos valores. Adicionalmente é também possível uma comunicação direta com um AR, uma vez obtida sua identificação.

Com tanta informação sendo compartilhada faz-se necessário a agregação da informação de contexto em uma entidade única, chamada, no sistema, de Interpretador de Contexto (IC). O IC é capaz de agrupar diversas VCs, organizadas em uma expressão lógica. Sendo também um AR, o IC deve ser subscrito para que se saiba da validação da expressão lógica. Na "computação calma" não só há uma maior integração entre os recursos, mas há também uma reação à mudança do contexto, portanto, os ARs atuadores devem se subscrever em ICs para desempenharem ações no meio.

Um dos focos deste trabalho é a simplicidade no desenvolvimento das aplicações, onde o esforço do desenvolvedor é minimizado. Um outro foco é na declaração das preferências por parte do usuário final. Junto do protótipo está sendo contruída uma interface gráfica que possibilita a usuários não expecializados definir regras para serem avaliadas no sistema, caracterizando um sistema de programação por usuários finais (*end-user programming*).

2. Trabalhos Relacionados

3. Conceitos Básicos

3.1. Prototipagem de Aplicações Pervasivas

4. Proposta do Framework

4.1. Modelo de Componentes Distribuídos

4.2. Comunicação

4.3. Suporte ao Gerenciamento de Recursos

Figura: Comunicação direta e indireta em alto nível

4.4. Variáveis de Contexto e Operações

Figura: Arquitetura que mostre as portas de entrada e saída dos ARs (VCs e Operações). Deve mostrar, ou ser explicado como se ligar.

4.5. Interpretador de Contexto

Figura: Arquitetura em camadas: desde os recursos até a aplicação, passando pelo IC

4.5.1. Criação de Interpretador de Contexto

Figura: Figura que demonstre essa rotina.

4.5.2. Interpretador de Contexto em Funcionamento

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

4.5.3. Temporização

Figura: Figura equivalente à anterior, mas mostrando com o temporizador.

4.6. Interface Gráfica de Composição de Regras

Se tiver alguma coisa da GUI pronta até lá, colocar uma figura aqui.

5. SmartAndroid

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.

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.

Coulouris, G., Dollimore, J., and Kindberg, T. (2005). *Distributed systems: concepts and design*. Addison-Wesley Longman.

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.

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.

Lyytinen, K. and Yoo, Y. (2002). Ubiquitous computing. *COMMUNICATIONSOF THE ACM*, 45(12):63.

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.

Saha, D. and Mukherjee, A. (2003). Pervasive computing: a paradigm for the 21st century. *Computer*, 36(3):25–31.

Satyanarayanan, M. (2001). Pervasive computing: Vision and challenges. *Personal Communications, IEEE*, 8(4):10–17.

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.