Linguagens de Programação para a Computação Pervasiva

Douglas Pereira Pasqualin[1], **Juliana Kaizer Vizzotto**[1], **Giovani Rubert Librelotto**[1], **André Rauber Du Bois**[2]

[1]Programa de Pós-Graduação em Informática – Universidade Federal de Santa Maria
Avenida Roraima, 1000 – 91.105-900 – Santa Maria – RS

[2]Universidade Católica de Pelotas (UFPel)
Rua Felix da Cunha, 412 – 96.010-000 – Pelotas – RS

{dpasqualin,juvizzotto,librelotto}@inf.ufsm.br, dubois@ufpel.edu.br

1. Introdução

A computação pervasiva define uma importante evolução da computação, que se iniciou na década de 70, quando os primeiros computadores para uso pessoal começaram a ser produzidos [Saha and Mukherjee 2003].

Sistemas pervasivos devem ter a capacidade de reconhecimento automático de contexto, ou seja, as aplicações pervasivas devem adaptar-se conforme o contexto em que estão inseridas [Rarau et al. 2007]. O reconhecimento de contexto é muito importante, pois segundo [Saha and Mukherjee 2003] um dos conceitos da computação pervasiva é a invisibilidade. Por exemplo, o usuário deve interagir naturalmente com o sistema, como se a tecnologia fizesse parte do ambiente, devendo ser evitadas ao máximo as intervenções do usuário para o funcionamento do sistema.

Segundo [Henricksen and Indulska 2004], a computação pervasiva cria uma série de novos desafios, principalmente quando comparada à computação atual, voltada para computadores pessoais. Um desses desafios é o reconhecimento de contexto, mais especificamente como desenvolver aplicações que consigam perceber o ambiente, conseguindo extrair informações e agindo pró-ativamente a favor do usuário. Nesse sentido, novas linguagens ou até mesmo novos paradigmas de programação estão sendo desenvolvidos, tentando tornar mais intuitiva a programação de aplicações pervasivas.

Este resumo tem o objetivo de apresentar brevemente como algumas das principais linguagens de programação propostas na literatura tratam os novos desafios da computação pervasiva.

2. Linguagens de Programação Pervasivas

A primeira alternativa pesquisada é um novo paradigma de programação, uma alternativa ao atual paradigma orientado a objetos. Torben Weis [Weis et al. 2006] apresenta o paradigma de programação baseado no Cálculo de Ambientes (*Ambient Calculus*). Para ele, os elementos do mundo real podem ser tratados como um ambiente (ao invés de objetos). Um ambiente é um espaço delimitado, onde acontece a computação. Um cenário mais complexo, como por exemplo, uma reunião acontecendo com vários participantes, compartilhando vários documentos, pode ser modelada como um Cálculo de Ambiente. O autor acredita que o Cálculo de Ambientes pode ser usado como uma base teórica para linguagens de programação sensíveis ao contexto. Por fim, ele apresenta o protótipo de uma nova linguagem de programação que implementa esses novos conceitos: a *N#*.

Outro paradigma denominado *Multifaceted Programming* é proposto por Anca Rarau [Rarau et al. 2007]. Para os autores, qualquer serviço que possua reconhecimento de contexto deve possuir dois tipos de comportamento: totalmente definido (controlado pelas facetas) e determinístico (controlado pela estratégia de exposição). O contexto associado a uma faceta se comporta como um interruptor. A adaptação do contexto de uma entidade multifacetada é dada pela troca de facetas. Uma entidade de programação multifacetada pode conter qualquer número de facetas, porém somente uma poderá ser exposta por vez. Para testar essa nova teoria, os autores estenderam a linguagem *C#*, criando assim uma nova linguagem denominada *AwareC#*.

Zhang Junbin [Junbin et al. 2009] argumenta em seu trabalho, que no paradigma de *Multifaceted Programming* todas as facetas devem ser definidas no momento de criação da aplicação, e não podem ser adicionadas e nem modificadas em tempo de execução. Assim, a proposta dos autores é utilizar uma nova alternativa denominada programação *table-driver*. A idéia principal é utilizar tabelas virtuais, que são criadas em tempo de compilação e alimentadas com novas informações pelo gerenciador de espaços em tempo de execução. Os valores contidos nas tabelas virtuais são utilizados pelas entidades *table-driver*, ou seja, as variáveis e funções da aplicação. Os autores acreditam que as tabelas virtuais isolam a complexidade de operações relacionadas com o contexto, simplificando o processo de desenvolvimento.

3. Considerações Finais

Conclui-se que o desenvolvimento de linguagens para computação pervasiva, é um assunto atual e desafiador, pois necessita levar em consideração novas características ímpares, tais como mobilidade e sensibilidade ao contexto.

Nesse contexto, esse trabalho de mestrado busca contribuir para o desenvolvimento de uma linguagem de programação para computação pervasiva de alto nível, tendo como foco a simplicidade de código, através da investigação de fundamentos semânticos para programação pervasiva.

Referências

Henricksen, K. and Indulska, J. (2004). A software engineering framework for context-aware pervasive computing. In *Proceedings of the 2nd IEEE Conference on Pervasive Computing and Communications (PerCom)*, pages 77–86, Orlando, Florida, USA.

Junbin, Z., Yong, Q., Di, H., and Ming, L. (2009). A table-driven programming paradigm for context-aware application development. *IEEE/IPSJ International Symposium on Applications and the Internet*, pages 121–124.

Rarau, A., Benta, K. I., and Cremene, M. (2007). Multifaceted based language for pervasive services with deterministic and fully defined behavior. *International Conference on Pervasive Services*, pages 76–79.

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

Weis, T., Becker, C., and Brändle, A. (2006). Towards a programming paradigm for pervasive applications based on the ambient calculus. In *International Workshop on Combining Theory and Systems Building in Pervasive Computing, co-located with Pervasive 2006*, Dublin, Ireland.