Gerência de Infra-estrutura Distribuída na Arquitetura de Desenvolvimento de Aplicações Ubíquas Continuum

Rodolfo Stoffel Antunes[1], **Cristiano André da Costa**[1]

[1]UNISINOS – Universidade do Vale do Rio dos Sinos
Av. Unisinos, 950 – São Leopoldo, RS

rodolfo.s.antunes@gmail.com, cac@unisinos.br

Resumo. *O avanço da computação ubíqua traz a necessidade do projeto de novas e complexas aplicações. Uma infra-estrutura de software permite a abstração de diversas questões de implementação, simplificando seu desenvolvimento. A infra-estrutura de software Continuum foi projetada de acordo com as principais questões de pesquisa da computação ubíqua. Este artigo apresenta a pesquisa desenvolvida durante o projeto dos serviços cell information base e communicator, utilizados pelo Continuum.*

Abstract. *The advances in Ubiquitous Computing brings the need to design new and complex applications. A software infrastructure can be used to abstract many implementation questions, simplifying their development. The Continuum software infrastructure was designed considering the main research topics in ubiquitous computing. This paper presents the research carried on during the design of the services cell information base and communicator, used by Continuum.*

1. Introdução

O termo computação ubíqua foi utilizado pela primeira vez em [Weiser 1991]. O autor afirma que é necessária uma maior integração da tecnologia da informação com o cotidiano, de forma que ela se torne transparente, auxiliando os usuários a lidar com a sobrecarga diária de informações. Pesquisas na computação ubíqua englobam questões de diversas áreas, dentre elas sistemas distribuídos e computação móvel, além de questões próprias da computação pervasiva [Satyanarayanan 2001].

Um estudo sobre os desafios que caracterizam a computação ubíqua é apresentado em [Costa et al. 2008]. Este estudo é utilizado na definição de um modelo abrangente de infra-estrutura de software para a computação ubíqua, que tem como elementos principais um *framework*, para auxiliar no projeto de aplicações para a arquitetura, e um *middleware*, responsável por tratar diversas questões da execução destas aplicações.

Este modelo é a base da definição do Continuum [Costa 2008], uma infra-estrutura de software para aplicações ubíquas, baseada em uma arquitetura orientada a serviços (SOA). Uma SOA é composta por um conjunto de serviços base, que podem ser utilizados para a criação de novos serviços e aplicações através de composição e gerenciamento [Papazoglou and Georgakopoulos 2003]. Assim como no modelo abrangente, o Continuum é composto por um *framework* para auxílio ao desenvolvimento de aplicações, e um *middleware*, que tem como núcleo uma série de serviços plugáveis, responsáveis pelas principais funcionalidades do ambiente de execução.

Atualmente, há apenas um modelo da interface dos serviços que compõe o *middleware* do Continuum, sendo necessária sua modelagem e implementação. Este artigo apresenta as pesquisas realizadas para determinar os requisitos e possíveis metodologias a serem empregados no projeto de dois serviços utilizados pelo Continuum: o *cell information base*, responsável pela manutenção do ambiente distribuído formado pelas entidades que executam o Continuum; e, *communicator*, que provê um mecanismo de comunicação por eventos para a infra-estrutura.

A seção 2 apresenta o modelo de abstração de entidades físicas do Continuum e a pesquisa realizada para a definição das propriedades do serviço *cell information base*. A seção 3, por sua vez, apresenta uma análise do estado da arte na área de sistemas de comunicação por eventos, que é a base do serviço *communicator*. Finalmente, na seção 4, são apresentas considerações finais relativas ao trabalho desenvolvido e sua continuidade.

2. Manutenção de Infra-estrutura

Em [Costa 2008] é proposto um modelo de abstração que permite a representação, no contexto de aplicações ubíquas, de entidades relevantes do mundo real. No modelo, uma *CoDimension* engloba todas as entidades que podem estar contidas no ambiente de uma aplicação, sendo composta por um conjunto de células, denominadas *CoCells*, que representam os lugares do ambiente modelado. Uma célula pode, ainda, ser composta por outras células, possibilitando a representação de lugares compostos. O nível de abstração representado por uma célula varia com a aplicação desenvolvida. *CoPersons* representam as pessoas presentes no ambiente, que estão localizadas em uma célula, e podem se deslocar para outras. Os objetos representados pelo modelo, denominados *CoNodes*, englobam dispositivos eletrônicos presentes nas células, como computadores, sensores, ou dispositivos móveis. CoNodes representam dispositivos estáticos básicos, tais como computadores de mesa. Além desses nodos, existem tipos particulares de representações empregadas na infra-estrutura. Cada célula possui um nodo especial, denominado *CoBase*, responsável pela execução de serviços básicos para seu gerenciamento. Dispositivos com características móveis, como um notebook com conexão por rede sem fio, são denominados *CoMobis*. Dispositivos móveis com propósitos específicos, como celulares ou PDAs, são tratados de maneira diferenciada pelo ambiente e são denominados *CoGadgets*.

A manutenção da infra-estrutura modelada é gerenciada pelo serviço *cell information base*, que também permite a consulta aos recursos presentes no ambiente. Esta funcionalidade pode ser comparada a de um serviço de gerenciamento de informações de contexto, porém estes serviços têm uma abrangência maior no escopo das informações coletadas, o que não é objetivo do CIB. Para definir-se os requisitos para a modelagem do CIB, foram consultadas referências a serviços similares, presentes em outras infra-estruturas para a computação ubíqua, dentre elas o Aura [Garlan et al. 2002], o Gaia [Roman et al. 2002], e o EXEHDA [Yamin 2004].

Os serviços pesquisados atuam como centralizadores, responsáveis pelas informações sobre o contexto das aplicações. A forma de coleta varia com o tipo dos dados armazenados pelo serviço. No caso do registro de entidades como dispositivos e usuários presentes no ambiente, os dados são informados diretamente pelas entidades registradas. Outras informações, por exemplo o ponto geográfico em que a entidade está localizada, são obtidas de outros sensores no ambiente, que operam em conjunto com o serviço.

Os serviços utilizam, para persistir os dados adquiridos, bancos de dados relaci-

onais ou serviços de diretório. A persistência pode ser realizada por um serviço centralizado, que se torna responsável por todas as resquisições do ambiente. Outros serviços utilizam persistência distribuída, onde cada instância é responsável por armazenar uma porção dos dados do ambiente, e as consultas são realizadas através da propagação de mensagens na rede, o que pode criar sobrecargas durante as consultas.

Para a realização de pesquisas, alguns sistemas utilizam funções especiais, que recuperam uma entidade e todos seus atributos, a partir de informações como o nome ou identificador de uma entidade. Outros serviços usam uma linguagem com uma semântica similar à da SQL, que é interpretada pelo serviço, permitindo maior flexibilidade na seleção e filtragem dos atributos retornados. Outro método é o envio de consultas diretamente na linguagem do sistema de persistência utilizado pelo serviço.

3. Comunicação Orientada a Eventos

O serviço *communicator* implementa um sistema de eventos para o Continuum, baseado no modelo de comunicação publicar-assinar (*publish-subscribe*). [Eugster et al. 2003] afirma que a comunicação orientada a eventos permite que provedores e assinantes sejam desacoplados nas dimensões de espaço, tempo e sincronização. Provedores e assinantes não mantêm contato direto, utilizando o serviço de notificações para comunicação. Além disso, não é necessário que provedores e assinantes estejam ativos simultaneamente para que a interação seja realizada. Por fim, operações de publicação e recepção de eventos são métodos não-bloqueantes, o que permite aos nodos mantêrem suas atividades durante a comunicação, que ocorre de forma assíncrona. A anonimidade e assincronismo inerentes deste modelo o tornam ideal para sistemas distribuídos [Huang and Garcia-Molina 2004]. Para definir os requisitos do serviço *communicator*, foram estudados outros sistemas publicar-assinar, dentre eles o Hermes [Pietzuch and Bacon 2002], o Echo [Eisenhauer et al. 2006], e o *Java Messaging Services* [Hapner et al. 2002].

Nos sistemas estudados, a separação entre provedores e clientes é realizada por uma entidade centralizadora. Trabalhos com foco em alto desempenho e adaptabilidade usam como base uma rede de *overlay*, para tratar o roteamento eficiente das notificações. Neste caso, a entidade central é implementada de maneira distribuída, e é executada pelos pares que utilizam o sistema de eventos. Estes sistemas apresentam escalabilidade e adaptabilidade, mas sobrecarregam os clientes que utilizam o sistema.

Alguns dos sistemas permitem que os eventos sejam temporariamente armazenados, de forma que a desconexão de clientes não ocasione a perda de notificações. Esta funcionalidade é limitada pelo espaço disponível na entidade central que gerencia os eventos, que se for totalmente distribuída, vai depender do armazenamento disponível nos clientes.

O método utilizado na definição das assinaturas pode ser baseado em um sistema de tipos, onde clientes especificam os eventos que desejam receber a partir dos tipos de dados contidos nas notificações. Este mecanismo reduz as conversões de dados durante o tráfego de notificações. O sistema, porém, torna-se dependente de uma linguagem de programação ou biblioteca de tipos. O uso de um sistema de tópicos, onde assinaturas são definidas a partir de tópicos pré-definidos no sistema, elimina esta restrição e mantém o desempenho do sistema, porém apresenta pouca flexibilidade para a especificação de assinaturas. Assinaturas por conteúdo, definidas a partir dos atributos presentes na própria notificação, através de uma linguagem de pesquisa, apresentam maior flexibilidade para a filtragem de eventos, porém uma maior sobrecarga no tratamento das assinaturas.

4. Considerações Finais

A computação ubíqua tem avançado rapidamente, graças à popularização dos dispositivos móveis, aliado a capacidade desses de acesso a redes sem fio. O desenvolvimento de uma aplicação ubíqua, por sua vez, tem como desafio o tratamento das diversas questões relacionadas com a área, tornando seu projeto uma tarefa complexa. Tendo como base um modelo atualizado para a computação ubíqua, a infra-estrutura de software Continuum apresenta diversas vantagens para o desenvolvimento destas aplicações.

Este artigo apresentou a pesquisa realizada com o objetivo de definir os requisitos para os serviços *cell information base* e *communicator*, presentes no *middleware* do Continuum. Dando continuidade ao trabalho, será realizado o detalhamento do projeto destes serviços, e em seguida sua implementação, com o objetivo de integrá-los ao Continuum, que atualmente encontra-se em desenvolvimento.

Referências

Costa, C. (2008). *Continuum: A Context-aware Service-based Software Infrastructure for Ubiquitous Computing*. PhD thesis, UFRGS.

Costa, C., Yamin, A., and Geyer, C. (2008). Towards a general software infrastructure for ubiquitous computing. *IEEE Pervasive Computing*, 7(1):64–73.

Eisenhauer, G., Schwan, K., and Bustamante, F. (2006). Publish-subscribe for high-performance computing. *IEEE Internet Computing*, 10(1):40–47.

Eugster, P., Felber, P., Guerraoui, R., and Kermarrec, A. (2003). The many faces of publish/subscribe. *ACM Computing Surveys*, 35(2):114–131.

Garlan, D., Siewiorek, D., Smailagic, A., and Steenkiste, P. (2002). Project aura: Toward distraction-free pervasive computing. *IEEE Pervasive Computing*, 1(3):22–31.

Hapner, M., Burridge, R., Sharma, R., Fialli, J., and Stout, K. (2002). Java message service. Disponível em <http://java.sun.com/products/jms/docs.html>. Acesso em Out. 2008.

Huang, Y. and Garcia-Molina, H. (2004). Publish/subscribe in a mobile environment. *Wireless Networks*, 10(6):643–652.

Papazoglou, M. and Georgakopoulos, D. (2003). Introduction: Service-oriented computing. *Communications of the ACM*, 46(10):24–28.

Pietzuch, P. and Bacon, J. (2002). Hermes: A distributed event-based middleware architecture. In *22nd International Conference on Distributed Computing Systems Workshops, 2002, Vienna*, pages 611–618, Washington. IEEE.

Roman, M., Hess, C., Cerqueira, R., Rananathan, A., Campbell, R., and Nahrstedt, K. (2002). A middleware infrastructure for active spaces. *IEEE Pervasive Computing*, 1(4):74–83.

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

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

Yamin, A. (2004). *Arquitetura para um Ambiente de Grade Computacional Direcionada às Aplicações Distribuídas, Móveis e Conscientes de Contexto da Computação Pervasiva*. PhD thesis, UFRGS.