CAUS: Uma arquitetura sensível ao contexto e orientada a componentes para sistemas de administração de ensino

Windson Viana[1,2], José Bringel Filho[2], José Celso F. Junior[3], Galeno J. de Sena[3], Edson L. F. Senne[3], Marlène Villanova Oliver[2], Jérôme Gensel[2], Hervé Martin[2]

[2]Laboratoire d'Informatique de Grenoble, Equipe STEAMER

[3]Universidade Estadual Paulista (UNESP) – Campus de Guaratinguetá, Departamentos de Engenharia Elétrica e Matemática

{jcfreire@reitoria, gsena@feg, elfsenne@feg}.unesp.br, {Windson.Viana-de-Carvalho, Bringel, Jerome.Gensel, Marlene.Villanova-Oliver, Herve.Martin}@imag.fr

Abstract. *In the scenario of Web systems, in particular applied to education administration systems, the increasing use of mobile devices and wireless technologies enabled the emergence of context-aware services. However, the development of these services is not trivial mainly due to the heterogeneity of mobile devices, the management of context information, and the complexity of the adaptation process. With the objective of assisting the development of these services, this paper proposes a context-aware component-based architecture called CAUS.*

Resumo. *No cenário dos sistemas Web, em particular, dos sistemas de administração de ensino, a crescente utilização de dispositivos móveis e das tecnologias de comunicação sem fio proporcionou surgimento de novos serviços chamados sensíveis ao contexto. Entretanto, o desenvolvimento de tais serviços impõe desafios principalmente em decorrência da heterogeneidade dos dispositivos moveis, da gestão das informações de contexto e da complexidade de implementação dos mecanismos de adaptação. Com o objetivo de auxiliar o desenvolvimento destes serviços, este artigo propõe uma arquitetura sensível ao contexto e orientada a componente chamada CAUS.*

1. Introdução

A larga utilização de dispositivos móveis (DM) e a evolução das tecnologias de comunicação sem fio ampliaram as formas de acesso aos sistemas Web atuais [Ramos 2005], dentre os quais se destacam os destinados à aprendizagem e à administração de ensino. Esses sistemas são compostos por aplicações que buscam facilitar/auxiliar as atividades administrativas e o acompanhamento pedagógico em ambientes escolares e universitários [Faure 2006]. A possibilidade oferecida pelos DM de obter acesso a serviços e a informações a qualquer momento e em qualquer lugar favorece o surgimento dos chamados serviços sensíveis ao contexto (*context-aware services*). Os DM, na sua grande maioria, apresentam limitações de recursos (e.g. tamanho de tela reduzida, baixa capacidade de processamento e memória) que restringem o ambiente de

[1] Bolsista de doutorado da CAPES

execução de aplicações móveis (i.e. aplicações desenvolvidas para estes dispositivos). Em contrapartida, eles possuem um número crescente de sensores (e.g. câmera, *Bluetooth*, receptor GPS) e de fontes de dados (e.g. agenda, contatos, redes sociais) que podem ser combinados para descrever a situação do usuário (e.g., localização, atividade) no momento em que este realiza o acesso ao sistema. Uma vez estabelecido o *contexto de utilização*, o sistema Web será capaz de fornecer serviços e conteúdo de forma adaptada à situação corrente do usuário [Ramos 2005, Henricksen 2004].

Apesar da intensa pesquisa relacionada aos sistemas sensíveis ao contexto realizada nos últimos anos [Baldauf 2007], o desenvolvimento e a implantação de tais sistemas ainda corresponde a uma operação complexa, que é resultante principalmente dos três pontos críticos a seguir: *i) a heterogeneidade dos DM de acesso; ii) a dificuldade em capturar/gerir as informações que descrevem o contexto de utilização; e iii) a complexidade relacionada à implementação dos mecanismos de adaptação e de filtragem de conteúdo baseados no contexto.*

Este artigo propõe uma arquitetura orientada a componentes que busca reduzir a complexidade de desenvolvimento resultante das características enunciadas acima. A arquitetura fornece mecanismos de adaptação, tanto no momento da instalação quanto durante a execução, de funcionalidades, conteúdo e interfaces de sistemas Web de administração de ensino em função do *contexto de utilização* do sistema. O restante do artigo está organizado como a seguir: a Seção 2 apresenta os trabalhos relacionados aos sistemas e arquiteturas sensíveis ao contexto; a Seção 3 apresenta a arquitetura proposta; a Seção 4 descreve o estudo de caso que demonstra a integração da arquitetura ao sistema SW3A (Sistema Web para Avaliação e Acompanhamento Acadêmico) da Universidade Estadual Paulista (UNESP). Por fim, a Seção 5 apresenta as considerações finais e perspectivas de trabalhos futuros.

2. Sistemas Sensíveis ao Contexto

Os sistemas sensíveis ao contexto exploram as informações que descrevem a situação do usuário com o objetivo de adaptar seus serviços e trazer benefícios à usabilidade e ao desempenho do sistema. Dey [Dey 2001] define contexto como "*qualquer informação que pode ser utilizada para caracterizar a situação de um elemento relevante para a interação entre o usuário e o sistema, incluindo, como elementos, o usuário e o próprio sistema*". Os elementos que irão compor o contexto dependem da sua relevância para o sistema em conhecê-los e da possibilidade em observá-los.

As informações que descrevem o contexto de utilização são utilizadas principalmente nos seguintes tipos de aplicação: *i) recomendação de serviços e conteúdo*, e.g. o sistema proposto em [Freyne 2007] envia as mensagens (SMS, MMS) de acordo com o contexto do destinatário (e.g. somente usuários presentes no auditório); ii) *adaptação do comportamento do sistema,* e.g. CRUMPET [Zipf 2002] modifica o estilo gráfico de mapas visualizados a partir de PDA em função da posição do usuário e do seu perfil de visualização; e *iii) anotação automática de multimídia,* e.g. PhotoMap [Viana 2008] é uma aplicação de anotação capaz de identificar o contexto do usuário (e.g. localização, pessoas próximas) no momento da captura de uma foto utilizando um PDA e utilizá-lo para a geração automática de "tags" de consulta. Nas instituições de ensino, os sistemas sensíveis ao contexto existentes são utilizados, em sua maioria, no auxílio ao aprendizado através de técnicas de M-Learning [Marçal, 2005]. Existem

ainda sistemas que não se enquadram nos tipos citados, tais como o proposto em [Faure, 2006], que utiliza um PDA com uma conexão Wifi para auxiliar os alunos na locomoção interna no campus universitário e para enviar mensagens informativas.

Esses sistemas consideram, em grande parte, um grupo restrito de elementos contextuais (e.g. localização e tempo) durante a adaptação e normalmente são desenvolvidos para atender a requisitos específicos de uma aplicação (e.g. auxílio à navegação). Portanto, a flexibilidade e a reutilização são características muitas vezes negligenciadas durante a concepção desses sistemas [Baldauf 2007]. Arquiteturas e *frameworks* de propósito gerais foram então propostos com o objetivo de implementar as principais funcionalidades de um sistema sensível ao contexto: aquisição das informações que descrevem o contexto; agregação e fusão dessas informações; interpretação e inferência do contexto; detecção de situações predefinidas; suporte e a execução dos mecanismos de adaptação [Henrickseen 2004]. COBRA [Chen 2004], COMMANTO [Strimpakou 2006] e MOCA [Sacramento 2004] são alguns exemplos de infra-estruturas que implementam essas funcionalidades. Porém, elas não são, em sua maioria, adequadas às aplicações móveis pois incorporam bibliotecas incompatíveis com muitas das plataformas de desenvolvimento para DM e apresentam também alto consumo de recursos [Preuvennes 2007]. Além disso, elas implementam a metodologia do tudo ou nada (*all-or-nothing*), i.e. Se o desenvolvedor deseja utilizar um serviço do *framework* (e.g. captura da localização), ele é obrigado a incorporar toda a arquitetura sensível ao contexto, mesmo que a aplicação não utilize a maioria dos outros serviços disponibilizados pela arquitetura.

3. Sensibilidade ao Contexto para Sistemas de Administração de Ensino

O CAUS *(Context-Aware University Software)* é uma arquitetura orientada a componentes para a captura/gestão de contexto e oferta de serviços. O CAUS está integrado ao sistema de administração de ensino chamado SW3A (veja Figura 1). O principal requisito desta arquitetura é oferecer a portabilidade entre a grande maioria de DM atuais, mesmo com um número reduzido de funcionalidades. Além disso, a captura e a fusão dos dados contextuais devem ser realizadas de forma a reduzir o consumo de recursos dos DM, considerando a heterogeneidade dos sensores existentes e a possibilidade de falhas/mudanças dos sensores durante a execução da aplicação. A Figura 1 ilustra os módulos que compõem a arquitetura proposta:

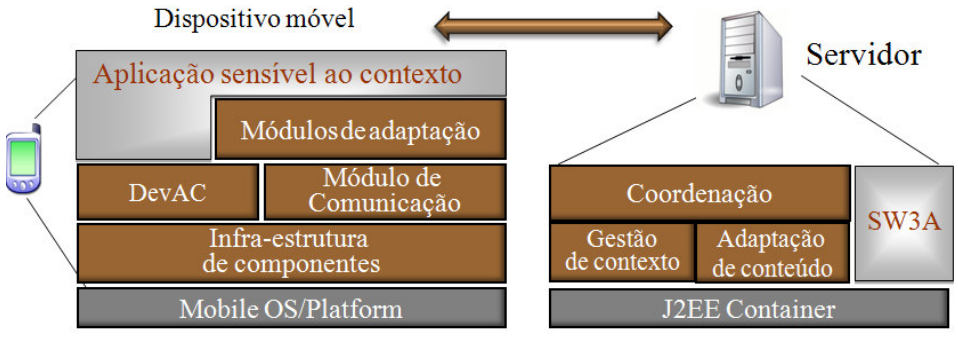

Figura 1- CAUS integrado ao sistema SW3A

- *Infra-estrutura de componentes:* principal módulo da arquitetura, responsável pelo suporte à **programação baseada em componentes**. O desenvolvimento de sistemas baseado em componentes oferece a separação das camadas de interface

e de implementação, que são as unidades de distribuição e implantação do sistema. A representação de aplicações como componentes inter-operáveis, além de promover a reusabilidade e reduzir a complexidade, permite a atualização de partes individuais de uma aplicação sem exigir, necessariamente, a interrupção da mesma. Esta característica, chamada de reconfiguração dinâmica, é a base para a implementação do mecanismo de troca de componentes em função do contexto (veja a Seção 3.1);

- *DevAC (Device Aware Context Toolkit):* módulo de aquisição de contexto que executa no DM e interage com o módulo de gestão de contexto do servidor para inferir/derivar informações de alto nível sobre o contexto de utilização;
- *Módulo de comunicação:* fornece às aplicações e ao DevAC serviços de envio/recepção de informações utilizando uma das tecnologias de comunicação suportadas pelo DM (e.g., Wifi, Bluetooth, GPRS);
- *Módulo de adaptação:* interage com o módulo de coordenação do servidor e a infra-estrutura de componentes do DM para realizar duas tarefas: a troca contextual dos componentes da aplicação e auto-adaptar os demais módulos da arquitetura em função do contexto computacional.

3.1. A infra-estrutura de componentes

A infra-estrutura de componentes foi inspirada na plataforma OSGi[2] e implementada em J2ME/MIDP[3]. Por um lado, a implementação realizada em J2ME/MIDP traz o beneficio da portabilidade, por outro lado, perde-se em funcionalidade. Por exemplo, a capacidade de realizar *download* e de instalar componentes em tempo de execução da aplicação não é disponível na nossa implementação, pois J2ME/MIDP permite apenas o *download* completo de uma aplicação ou de sua atualização. Mesmo com estas restrições, a adaptação em tempo de instalação e a troca sensível ao contexto dos componentes previamente instalados são suportadas pela infra-estrutura de componentes do CAUS.

Nessa infra-estrutura, um componente é chamado de *iLet* e representa uma unidade de implantação que pode compor uma aplicação. Um *iLet* não pode invocar métodos de outros *iLets* diretamente. A comunicação entre os *iLets* é realizada através de serviços que são publicados na infra-estrutura de componentes e somente são acessíveis através dela. Um serviço corresponde a uma *Interface Java* compartilhada entre o provedor e o cliente do serviço que são, eles próprios, *iLets* registrados na infra-estrutura. Um *iLet* é capaz de: i) executar código Java MIDP; ii) publicar serviços; iii) procurar/utilizar serviços disponíveis na infra-estrutura. Os *iLets* são distribuídos em pacotes JAR e devem conter, no mínimo, um arquivo *manifest* e uma classe que implementa a interface *iLetActivator*. O arquivo *manifest* descreve as dependências do *iLet*, os serviços utilizados e/ou fornecidos e o nome da classe executável do *iLet* (implementação da classe *iLetActivator*).

O mecanismo de troca de componentes sensível ao contexto é implementado na infra-estrutura usando o padrão de projeto *Business Delegate*[4]. No caso em que um serviço (i.e., uma interface Java) possui varias implementações e deseja-se ocultar dos clientes esta heterogeneidade, um mediador de serviço (*Service Delegate)* pode ser criado. Este novo componente contém uma classe que implementa a mesma interface e

[2] http://www.osgi.org/Main/HomePage
[3] http://java.sun.com/javame/index.jsp
[4] http://java.sun.com/blueprints/corej2eepatterns/Patterns/BusinessDelegate.html

possui acesso a todas as implementações disponíveis do serviço através da infra-estrutura. Um cliente em potencial poderá ter acesso ao serviço através do mediador que decidirá, dentre as existentes, qual é a implementação mais adequada naquele contexto. Esta decisão de adaptação pode ser predefinida no plano de implantação da aplicação ou codificada no interior do código do mediador. No primeiro caso, é o processo de implantação da aplicação que se encarregará de gerar o código do mediador a partir das informações do plano de implantação. A Figura 2 mostra um exemplo de mediador implementado no módulo de comunicação. Uma interface (serviço) de transmissão de informações DM-servidor foi definida para o CAUS. Duas implementações do serviço foram codificadas: a primeira, *Bluetooth Communicator*, utiliza-se de um ponto de acesso Bluetooth; e a segunda, *HttpConnection*, usa uma conexão Http tradicional (sobre uma conexão de rede GPRS ou WIFI). O Cliente X (um outro *iLet* do CAUS) que deseja enviar uma mensagem ao servidor irá acessar o mediador fornecido pelo *iLet Communicator* que decidirá qual é a implementação mais adequada. Através desta estratégia, permite-se a adição de novos componentes de comunicação e a eliminação de implementações que não podem executar no DM. No caso desse mediador, a decisão de adaptação em tempo de execução é baseada na detecção de um ponto de acesso Bluetooth. Caso ele não seja detectado, o componente *HttpConnection* é utilizado.

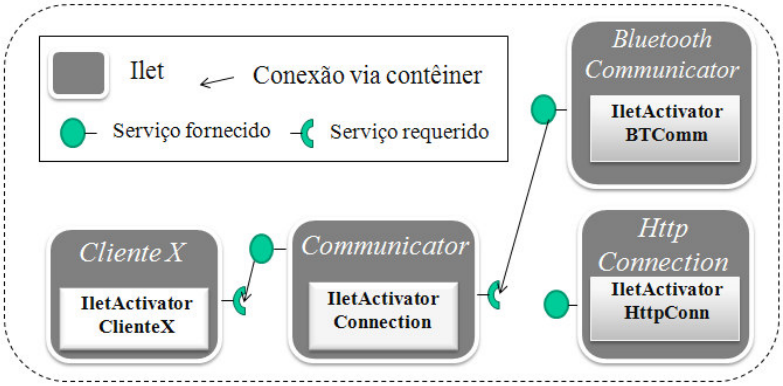

Figura 2- Exemplo de mediador : o iLet Communicator

A infra-estrutura fornece ainda um mecanismo de troca de estados entre as implementações, que é executado no momento em que o mediador realiza a modificação da implementação atual de um serviço. O *iLet* em execução é notificado pelo mediador sobre a sua iminente substituição. Este último, por sua vez, pode enviar, a uma região de memória compartilhada e gerida pelo contêiner, uma descrição do seu estado através de uma instância de *Object*. O novo iLet poderá acessar essa região e utilizar as informações de estado do componente anterior. O *manifest* do mediador contém uma descrição dos componentes gerenciados pelo serviço e a prioridade inicial de execução entre eles. Por exemplo, no caso do *Communicator*, a versão do componente de comunicação *Bluetooth* tem prioridade sobre a versão GPRS/WIFI.

Além da troca sensível ao contexto, a nossa arquitetura fornece a adaptação em tempo de instalação. Ao contrário de outras abordagens, o CAUS não exige a preinstalação de um *middleware* no DM. Apenas o suporte à plataforma J2ME MIDP e à navegação WAP[5] são necessários. A Figura 3 ilustra o processo de instalação proposto. O usuário, a partir de um navegador do DM, obtém acesso ao servidor e

[5] http://en.wikipedia.org/wiki/Wireless_Application_Protocol

escolhe a aplicação que ele deseja instalar. O servidor verifica o cabeçalho da requisição WAP e através de uma versão estendida do registro WURFL[6] identifica o perfil do DM utilizado pelo usuário. A versão estendida do registro contém informações detalhadas sobre o suporte dos pacotes adicionais da plataforma J2ME/MIDP, e.g. suporte a API JSR 82 (*Bluetooth*). Uma vez identificado o dispositivo, o plano de implantação da aplicação, i.e. o conjunto de *iLets* a serem instalados, é modificado a fim de conter os *iLets* nos quais as dependências de pacotes possam ser resolvidas. Por exemplo, se um componente utiliza a API JSR 82 e esta não é suportada pelo dispositivo, o *iLet* será removido do plano de implantação. Após a execução deste processo, o servidor verifica se a aplicação contém todos os *iLets* especificados como obrigatórios. Em caso positivo, um processo similar é disparado para selecionar os iLets da arquitetura que também será adaptada ao modelo do DM. Ao final do processo, um JAR contendo os próprios componentes da aplicação e os componentes da arquitetura é gerado e enviado ao DM. Mais detalhes sobre o processo de implantação são apresentados em [Viana 2009].

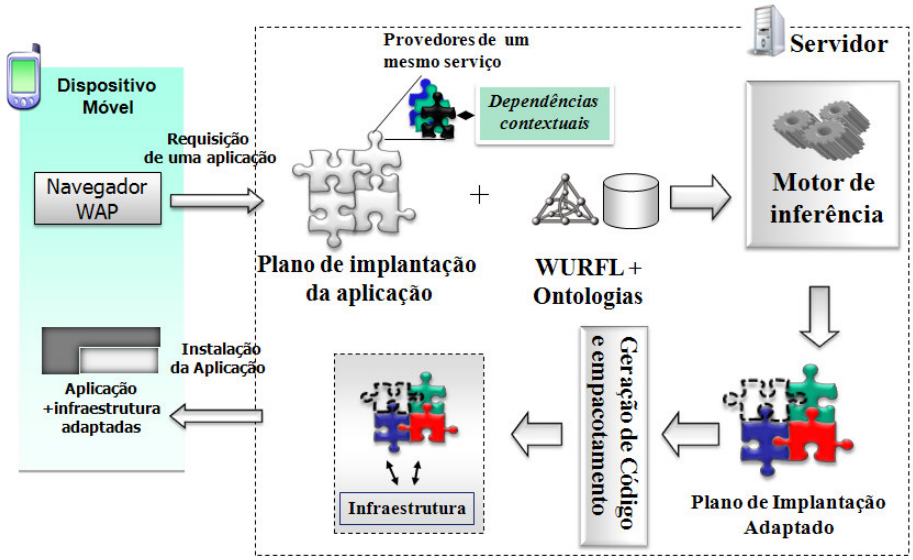

Figura 3 – Processo de adaptação da aplicação durante a instalação

3.2. DevAC – Device Aware Context Toolkit

DevAC foi inspirado na arquitetura de gestão de contexto proposta em [Henrickseen 2004]. Este módulo é um *middleware* orientado a componentes que possui acesso aos sensores existentes no DM (e.g. Câmera, GPS) e que realiza a fusão destas informações com os dados provenientes de outras fontes de informação (e.g. Web Services, a base de dados do SW3A). *DevAC* também monitora as características estáticas e dinâmicas da máquina virtual e do DM (e.g. os tipos de conexões de rede disponíveis).

O DevAC contém um serviço para disponibilizar informações contextuais que permite à aplicação ter acesso diretamente ao valor do sensor em modo síncrono (Figura 4). Por exemplo, a aplicação pode ter acesso aos endereços físicos dos dispositivos *Bluetooth* ao redor do usuário. Por outro lado, no modo assíncrono a aplicação registra no DevAC um intervalo de valores de um sensor (ou de sensores) que ela deseja ser notificada. Por exemplo, uma aplicação pode requisitar uma notificação caso um novo dispositivo *Bluetooth* tenha sido detectado. O DevAC oferece diversos formatos de

[6] http://wurfl.sourceforge.net/

exportação dos valores contextuais, tais como objetos JavaBeans[7], XML e instâncias da ontologia Context Top [Viana 2008].

Os sensores lógicos são implementados sobre a infra-estrutura de componentes, o que permite a existência de varias implementações para um mesmo sensor. Neste caso, um *ServiceDelegate* será criado, como acontece com o sensor de localização (Figura 3). Três componentes distintos foram desenvolvidos para capturar a localização do usuário: i) acesso direto ao GPS através da *LocationAPI* existente na plataforma J2ME; ii) *reverse geocoder*, que apresenta um formulário ao usuário solicitando o endereço ou o prédio em que ele se encontra; iii) localização via *Bluetooth/Wifi*, em que este componente consulta o componente de comunicação para descobrir o ponto de acesso de rede (Bluetooth ou Wifi) com o qual a aplicação cliente está conectada. Considera-se aqui que todos os pontos de acesso são geo-referenciados e que a localização do usuário (i.e. do DM) é igual à localização do ponto de acesso com o qual ele esta conectado. A partir do nome do ponto de acesso, este componente consulta um Web Service e recupera a coordenada GPS do ponto de acesso (e.g. o sistema SW3A mantem um cadastro de todos os pontos de acesso geo-referenciados na base de dados).

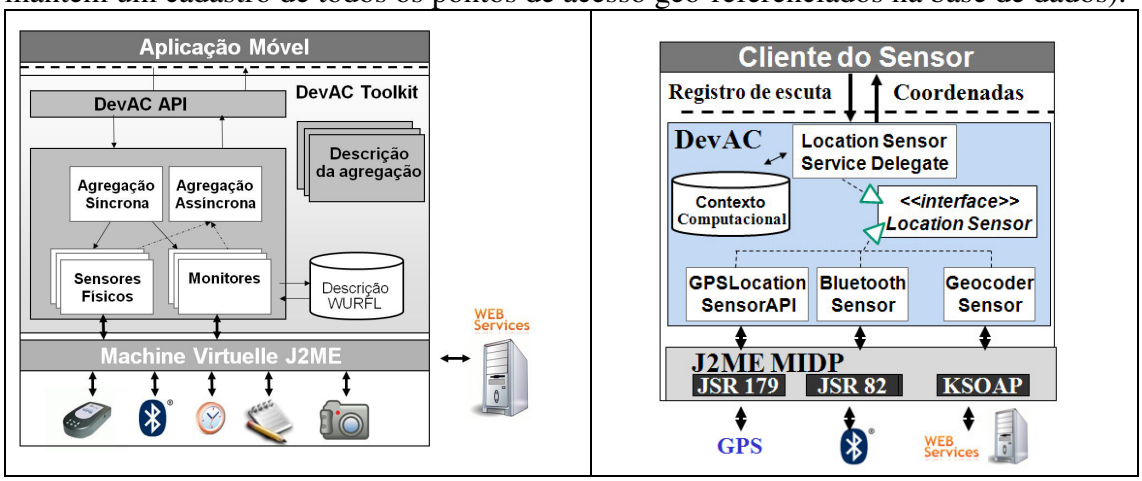

Figura 4 - DevAc e as implementações do sensores de localização

Os dois primeiros componentes de localização foram extraídos da aplicação PhotoMap [Viana 2008]. Eles implementam a interface de um sensor, *Location Sensor*, que envia notificações de atualização dos dados capturados aos cliente registrados para escutá-lo. Esses componentes retornam os dados capturados encapsulados em uma instância da classe *CoordinatesDataSensor*, que contém as coordenadas geográficas capturadas pelos sensores. Portanto, as três implementações do sensor de localização retornam as coordenadas geográficas (latitude, longitude, altitude) do DM.

4. Estudo de Caso

Serviços foram implementados para demonstrar a utilização do CAUS integrado ao sistema SW3A. Detalha-se a seguir o serviço *Context-Aware Messenger*.

CA Messenger permite o envio de mensagens sensíveis ao contexto aos usuários do SW3A. Através deste serviço, o coordenador do curso poderá, por exemplo, enviar uma mensagem de divulgação de um evento que será transmitida aos alunos localizados

[7] http://en.wikipedia.org/wiki/JavaBeans

nas proximidades do local. Ou ainda, notificar os alunos de uma disciplina que houve uma mudança da sala de aula. A Figura 5 apresenta um cenário de implantação do serviço em um prédio da UNESP[8]. Observa-se a presença de vários pontos de acesso de redes sem fio (WiFi e Bluetooth) geo-referenciados e distribuídos na infra-estrutura do prédio. Nesse cenário, o Aluno 1 deseja recuperar todas as mensagens válidas para o seu contexto de utilização (i.e. localização e tempo). Ele estabelece uma conexão com um ponto de acesso *Bluetooth* presente no corredor do prédio (seu DM não possui suporte a redes WiFi). O servidor detecta a sua localização e o seu identificador para, em seguida, enviar as mensagens cujo o contexto de notificação são satisfeitos pelo contexto atual do aluno. CA Messenger utiliza o CAUS para realizar as seguintes operações sensíveis ao contexto: adaptação do módulo de comunicação (instalação em tempo de execução), adaptação de conteúdo (e.g. filtro de mensagens), adaptação da apresentação (e.g. as mensagens podem ser formatadas e enviadas como mensagens internas ao serviço ou formatadas como um e-mail e enviadas ao usuário caso ele não esteja conectado no momento dos lembretes).

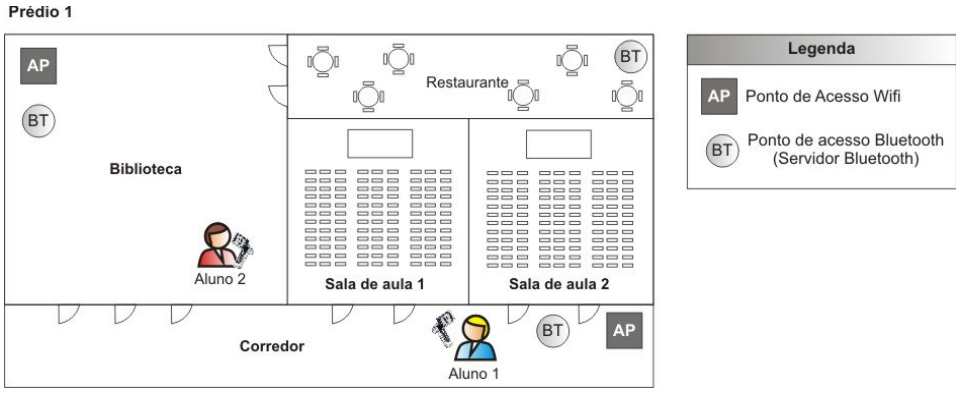

Figure 5. Cenário de implantação física para o serviço CA Messenger

O serviço é composto por dois módulos: o gestor de mensagens *CA Messenger Server* (servidor) e o *CA Message Viewer* (aplicação móvel). O *CA Messenger Server* contém uma interface de gestão de mensagens (cadastro, modificação, visualização, exclusão) ilustrada na Figura 6. Um coordenador/professor pode cadastrar uma mensagem, no qual é possível selecionar através de mapas (Google Maps API) a localização do evento e/ou o local onde ela será divulgada. Todos os locais de divulgação de mensagens (e.g. salas de aula, salas de seminário, auditórios, blocos, bibliotecas, salas de reunião) foram geometricamente representados e geo-referenciados na base de dados do SW3A (Postgres/*PostGIS*[9]). Um serviço foi codificado para estabelecer a localização semântica (e.g. sala X) à partir da localização simbólica do DM detectada pela DevAC (veja Seção 3.2) . O CA Messenger Server utiliza essa informação para detectar se uma regra de notificação de mensagem foi satisfeita.

O *CA Messenger Viewer* é uma aplicação J2ME instalada a partir do CAUS, que permite ao usuário recuperar as mensagens através de uma conexão com uma rede sem fio (GPRS/WIFI/Bluetooth). Os módulos de comunicação e de localização são adaptados no momento da instalação e podem ser trocados em tempo de execução pela infra-estrutura de componentes (veja Seção 3.1). Por exemplo, a detecção de um ponto

[8] Universidade Estadual Paulista (Campus de Guaratinguetá)

[9] http://postgis.refractions.net/

de acesso Bluetooth. A aplicação móvel permite também ao alunos recuperar as suas notas/freqüências, como ilustrado na Figura 7.

Figure 6. Exemplo de cadastro de mensagem e seleção da localização utilizando Google Maps.

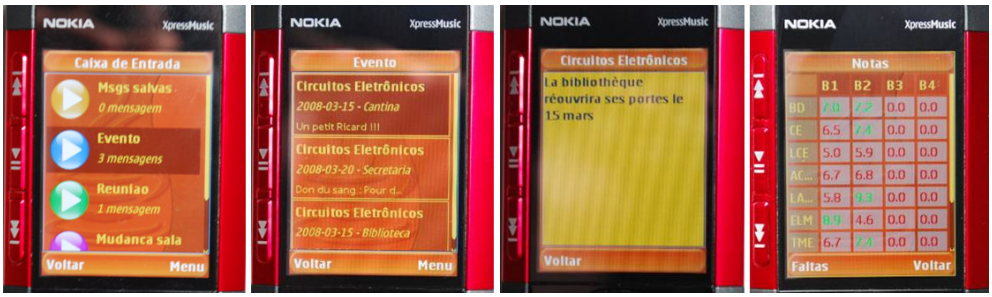

Figura 7. CA Messenger Viewer

5. Conclusão

Este artigo apresenta uma arquitetura orientada a componentes (CAUS) para o suporte de sistemas de administração de ensino sensíveis ao contexto. O CAUS oferece mecanismos de adaptação para os serviços do sistema e para a própria arquitetura em função do contexto computacional, e.g. tipo dispositivo, sistemas de comunicação disponíveis. A adaptação pode ocorrer, ao contrário de outras arquiteturas sensíveis ao contexto, tanto em tempo de instalação quanto em tempo de execução. Esta arquitetura foi desenvolvida em parceria pelas equipes do Brasil (UNESP) e da França (STEAMER-LIG) no âmbito do projeto FAPESP-CNRS n° 2004/08384-3. O CAUS foi incorporado ao sistema SW3A e serviços foram implementados (e.g. *Context-Aware Messenger*) com o objetivo de demonstrar a utilização da arquitetura e a viabilidade do sistema em um cenário real. Além disso, foi desenvolvido um módulo de comunicação *Bluetooth* para a troca de mensagens entre um DM e um ponto de acesso Bluetooth (servidor) que viabiliza a implantação dos serviços com baixo custo de instalação, operação e manutenção.

Como trabalhos futuros, pretende-se avaliar a aceitação dos serviços junto a alunos e professores das instituições de ensino envolvidas (Brasil e França), assim como propor extensões à arquitetura e novos serviços sensíveis ao contexto.

Bibliografia

Batista Junior, E. D., Delamaro, M. C., Ribeiro, R. M. S., Costa, A. F. B., Ribeiro, F. S. (2000) Acompanhamento da implantação de um curso de Engenharia de Produção: Sistema Integrado de Avaliação do Processo Ensino-Aprendizagem, In: Congresso Brasileiro de Ensino de Engenharia, Ouro Preto – MG.

Baldauf, M., Dustdar, S. and Rosenberg, F. (2007) "A survey on context-aware systems", Int. J. Ad Hoc and Ubiquitous Computing, Vol. 2, No. 4, pp.263–277.

Chen, H. (2004) "An Intelligent Broker Architecture for Pervasive Context-Aware Systems", PhD Thesis, University of Maryland, Baltimore County.

Dey, A.K. and Abowd, G.D., (2000). "Towards a Better Understanding of Context and Context-Awareness". HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing, Springer-Verlag, London, UK, 304-307.

Faure, C., Benci, P. Danzart, A., Lecolinet, E. (2006) Conception de services mobiles pour étudiants. UbiMob. Paris, France.

Freyne, J., Varga, E., Byrne, D., Smeaton, A. F., Smyth, B., Jones, G.J.F. (2007) "Realising context-sensitive mobile messaging". In: MONET, 25-30 November, Vilamoura, Portugal.

Henricksen K., Indulska J. (2004) "A Software Engineering Framework for Context-Aware Pervasive Computing". PerCom, Orlanda, USA, Mars, 77-86.

Marçal, E., Andrade, R., Rios, R. (2005) "Aprendizagem utilizando Dispositivos Móveis com Sistemas de Realidade Virtual", In: Revista Novas Tecnologias na Educação V. 3 Nº 1, Maio, Brasil.

Preuveneers, D. and Berbers, Y. (2007) "Towards context-aware and resource-driven self-adaptation for mobile handheld applications," Proceedings of the ACM symposium on Applied computing, Seoul, Korea, pp. 1165-1170.

Ramos, A. C., Gensel, J., Martin, H. (2005) "PUMAS: a Framework based on Ubiquitous Agents for Accessing Web Information Systems through Mobile Devices", 20th Annual ACM Symposium on Applied Computing - Web Technologies and Applications, Santa Fe, New Mexico, March 13 -17.

Sacramento, V. and Endler, M. (2004) "MoCA: A Middleware for Developing Collaborative Applications for Mobile Users" IEEE Distributed Systems Online, vol. 5, no. 10, pp. 2, Oct.

Strimpakou, M. A., Roussaki, I. G., Anagnostou, M. E., (2006) "A Context Ontology for Pervasive Service Provision". IEEE Conference on Advanced Information Networking and Applications (AINA'06).

Viana, W., Bringel Filho, J., Gensel, J., Villanova-Oliver, M., Martin, H. (2008) "PhotoMap: from location and time to context-aware photo annotations. Journal of Location Based Services", 2 (3), 211-235.

Viana, W.; Bringel Filho, J.; Villanova-Oliver, M.; Gensel, J.; Martin, H. (2009) "Aide au développement et au déploiement d'applications mobiles et sensibles au contexte : l'architecture CoMMediA". Actes de l'Atelier ERTSI. INFORSID 2009.

Zipf, A. (2002) "User-Adaptive Maps for Location-Based Services (LBS) for Tourism". ENTER Communications Technologies in Tourism, Innsbruck Austria.

Yoonhee K., Eun-kyung K., Jeuyoung K., Eunhye S., In-Young K. (2006) "Ontology Based Software Reconfiguration in a Ubiquitous Computing Environment". The Sixth IEEE International Conference on Computer and Information Technology, p. 260.