WebBEMS: um sistema baseado em componentes para o suporte à colaboração via Web

Cléver R. Guareis de Farias[1], Carlos E. Gonçalves[2], Marta Costa Rosatelli[2]

[1]Departamento de Física e Matemática
Universidade de São Paulo (FFCLRP/USP)
Av. Bandeirantes, 3900 – CEP 14040-901 – Ribeirão Preto – SP

[2]Programa de Mestrado em Informática
Universidade Católica de Santos (Unisantos)
R. Dr. Carvalho de Mendonça, 144 – CEP 11070-906 – Santos – SP

farias@ffclrp.usp.br, ceg-elus@unisantos.br, rosatelli@unisantos.br

Abstract. *Component-based development has been increasingly used in the development of collaborative services and systems. Software components allow solving several problems related to the distribution of collaborative systems. Besides, components can be configured, replaced and combined dynamically, increasing the flexibility, integration and adequacy of collaborative systems. This paper presents the development of an electronic meeting system named Web-Based Electronic Meeting System (WebBEMS), according to an architectural model conceived for the development of component-based collaborative systems. The paper depicts details of modelling, implementation and test of the application starting from the components architecture until its user interface.*

Resumo. *O desenvolvimento baseado em componentes tem sido cada vez mais utilizado no desenvolvimento de serviços e sistemas colaborativos. Os componentes de software possibilitam a resolução de vários problemas relacionados à distribuição dos sistemas colaborativos. Além disso, os componentes podem ser configurados, substituídos e combinados dinamicamente, aumentando dessa forma a flexibilidade, integração e adequação dos sistemas colaborativos. Este artigo apresenta o desenvolvimento de um sistema de reunião eletrônica, o Web-Based Electronic Meeting System (WebBEMS), segundo um modelo arquitetônico concebido para o desenvolvimento baseado em componentes de sistemas colaborativos. O artigo descreve em detalhes a modelagem, implementação e teste da aplicação partindo da arquitetura de componentes até a sua interface com o usuário.*

1. Introdução

O desenvolvimento baseado em componentes surgiu como paradigma de desenvolvimento de software no final dos anos 90. Este paradigma tem por princípio a construção de sistemas através da composição dos serviços individuais de um conjunto de componentes de software. Um componente de software é um pedaço binário de

software, auto contido, customizável e componível, com interfaces e dependências bem definidas [de Farias 2002].

O desenvolvimento baseado em componentes apresenta algumas vantagens se comparado ao desenvolvimento de software tradicional, tais como desenvolvimento das diversas partes que compõem um sistema de forma autônoma umas das outras, facilidade de instalação e manutenção de um sistema de software, e redução do tempo de desenvolvimento do sistema através do reuso.

Outra característica importante dos componentes de software é que os mesmos devem atender a uma especificação, que lhes conferem características importantes como compatibilidade e reusabilidade. Com a popularização dos componentes de software, surgiram várias especificações, desde as voltadas à componentização de interfaces gráficas, tais como as especificações *ActiveX*, *COM*, *VCL* e *JavaBeans*, até especificações mais complexas capazes de operar em um modelo de componentes distribuídos, tais como as especificações *CORBA*, *.NET* e *EJB*.

Componentes de software têm sido cada vez mais utilizados no desenvolvimento de serviços e sistemas colaborativos, c.f., [ter Hofte 1998, Brusilovsky 2004, Fuks *et al*. 2005, Pukkhem e Vatanawood 2005, de Farias *et al*. 2006]. Componentes são instalados e executados em plataformas distribuídas, contribuindo dessa forma para a resolução de vários problemas relacionados à distribuição dos sistemas colaborativos. Componentes também podem ser configurados, substituídos e combinados dinamicamente, aumentando dessa forma a flexibilidade, integração e adequação desses sistemas.

Um dos maiores desafios do desenvolvimento de um sistema baseado em componentes é a definição de sua arquitetura. O uso de um modelo arquitetônico concebido para o desenvolvimento baseado em componentes de sistemas colaborativos facilita não somente a definição da arquitetura deste sistema, mas também sua implementação e reuso dos componentes identificados. Neste sentido, este artigo apresenta o desenvolvimento de um sistema de reunião eletrônica, o sistema Web-Based Electronic Meeting System (WebBEMS), com base em um modelo arquitetônico projetado especificamente para atender sistemas colaborativos [de Farias 2005].

O restante deste artigo está estruturado da seguinte forma: a seção 2 apresenta uma visão geral do modelo arquitetônico utilizado neste trabalho; a seção 3 descreve os requisitos do WebBEMS, e detalhes da modelagem e arquitetura, mapeada para a tecnologia J2EE; a seção 4 descreve os detalhes de implementação e teste do WebBEMS; a seção 5 discute alguns trabalhos relacionados; finalmente na seção 6 são apresentadas algumas considerações finais e perspectivas futuras.

2. Modelo Arquitetônico

A arquitetura de um sistema computacional pode ser definida como a estrutura (ou estruturas) do sistema em termos de um conjunto de componentes, as partes visíveis desses componentes e seus relacionamentos [Bass *et al*. 1997]. Neste sentido, a arquitetura de um sistema pode ser vista como a decomposição de mais alto nível deste sistema em um conjunto de componentes, conjuntamente com uma caracterização de como estes componentes interagem [van Vliet 2000]. Um modelo arquitetônico define

os componentes básicos utilizados na construção de uma aplicação e o relacionamento entre estes componentes.

Nosso modelo arquitetônico define diferentes tipos de componentes que servem como base para o refinamento do sistema e dos componentes desse sistema. O modelo também define diferentes tipos de aspectos colaborativos que auxiliam na identificação dos diferentes tipos de componentes e de seus serviços [de Farias *et al.* 2005, de Farias *et al.* 2006]. Três diferentes tipos de componentes são identificados: *componente de aplicação, componente colaborativo e componente básicos.*

Um componente de aplicação corresponde a uma aplicação colaborativa, a qual pode ser utilizada separadamente ou integrada junto a outros componentes de aplicação. Um componente de aplicação consiste de um conjunto integrado de componentes colaborativos. Um componente colaborativo representa um dado conceito ou aspecto colaborativo, o qual pode ser reutilizado na construção de outros componentes colaborativos mais especializados ou diretamente na construção de componentes de aplicação. Um componente colaborativo consiste de um conjunto integrado de componentes básicos. Um componente básico é a unidade mais básica de projeto, implementação e implantação. Seu comportamento é o resultado de um único conjunto de código binário. Componentes básicos os são utilizados para tratar da distribuição dos aspectos e responsabilidades colaborativos de um componente colaborativo.

Os diferentes aspectos funcionais de um componente colaborativo podem ser agrupados em diferentes camadas, chamadas de camadas de aspectos colaborativos. Foram identificadas quatro diferentes camadas: *interface*, *usuário*, *colaboração* e *recurso*. Cada camada faz uso da funcionalidade provida por uma camada inferior para prover alguma funcionalidade utilizada por uma camada superior. Cada camada pode ser implementada por um ou mais componentes básicos.

A camada de interface está relacionada com a provisão de uma interface adequada entre um usuário humano e o componente colaborativo. Esta camada trata de toda comunicação direta com os usuários. A camada de usuário está relacionada com o suporte local às atividades executadas por um único usuário. Esta camada trata de questões locais relacionadas a cada usuário individualmente, não afetando a colaboração como um todo. A camada de usuário também relaciona a camada de interface com a camada de colaboração. A camada de colaboração está relacionada com a lógica de colaboração de múltiplos usuários. Esta camada é responsável pela implementação dos principais aspectos da colaboração e pela relação da camada do usuário com a camada de recurso. A camada de recurso está relacionada com o acesso às informações colaborativas compartilhadas, as quais são mantidas de forma persistente em um banco de dados, por exemplo.

Os aspectos de colaboração de uma camada podem estar ou não acoplados. O conceito de *acoplamento* foi introduzido como um mecanismo geral para unir as contribuições de um conjunto de usuários, de tal forma que estes poderiam compartilhar a mesma visão ou estado de uma colaboração [ter Hofte 1998].

Uma camada está acoplada se todos seus usuários possuem a mesma percepção das informações presentes na camada e de alterações nas mesmas. Quatro níveis de acoplamento podem ser estabelecidos baseando-se nas camadas definidas neste trabalho:

acoplamento de interface, *acoplamento de usuário*, *acoplamento de colaboração* e *acoplamento de recurso*.

O nível de acoplamento de interface representa o nível de acoplamento mais restrito. Todos os usuários do componente têm a mesma percepção da colaboração a partir da camada de interface. Este nível de acoplamento corresponde ao estilo de colaboração conhecido como What You See Is What I See (WYSIWIS).

O nível de acoplamento de usuário oferece mais liberdade para os usuários de um componente do que o nível de acoplamento de interface. Todos os usuários do componente possuem a mesma percepção da colaboração a partir da camada do usuário, i.e., as informações na camada de usuário são compartilhadas por todos os usuários, mas suas camadas de interface são mantidas independentemente umas das outras.

O nível de acoplamento de colaboração oferece um grau maior de liberdade do que o nível de acoplamento de usuário. Todos os usuários do componente têm a mesma percepção da colaboração a partir da camada de colaboração, i.e., as informações na camada de colaboração são compartilhadas por todos os usuários, mas suas camadas de interface e usuário são mantidas independentemente umas das outras.

O nível de acoplamento de recurso provê o maior grau de liberdade para a utilização do componente. Neste nível, todos os usuários do componente têm a mesma percepção da colaboração a partir da camada de recurso, i.e., as informações na camada de recurso são compartilhadas por todos os usuários, mas suas camadas de interface, usuário e colaboração são mantidas independentemente umas das outras.

A Figura 1 ilustra os diferentes níveis de acoplamento definidos neste trabalho para dois usuários. As figuras 1a, 1b, 1c e 1d mostram os níveis de acoplamento de interface, de usuário, de colaboração e de recurso, respectivamente.

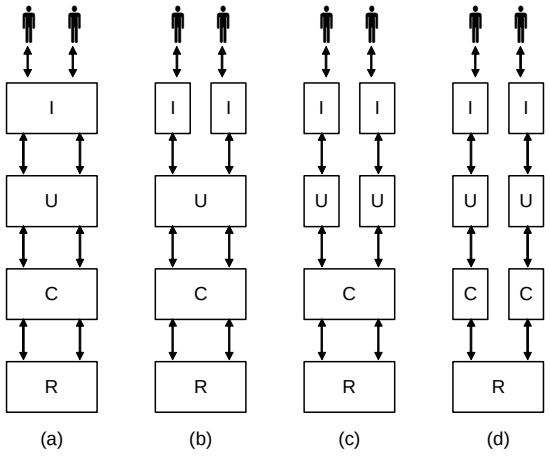

Figura 1. Níveis de acoplamento

3. Projeto do WebBEMS

3.1. Metodologia de desenvolvimento

O projeto utilizou uma metodologia baseada em um ciclo de desenvolvimento incremental e iterativo, fazendo uso da linguagem UML [OMG 2007a, OMG 2007b] para a modelagem do sistema. Em linhas gerais uma metodologia de ciclo incremental e iterativo propõe o desenvolvimento de sistemas de software através de iterações encadeadas, onde em cada iteração uma nova versão do sistema é produzida a partir do resultado obtido na iteração anterior acrescido das novas demandas de requisitos para a iteração atual.

Uma iteração ou ciclo de desenvolvimento é composto por diversas atividades de projeto. Em cada atividade de projeto um grupo de tarefas bem definidas deve ser executado com a finalidade de produzir e/ou alterar um ou mais artefatos, sejam estes documentos, modelos ou códigos. Uma metodologia de ciclo incremental e iterativo normalmente é composta por quatro atividades básicas: especificação de requisitos, análise, implementação e teste.

O projeto do WebBEMS também aplicou algumas práticas descritas pela metodologia *Extreme Programming* [Jeffries *et al.* 2000], referentes a temas como planejamento de versões, testes de unidade, testes de aceitação e integração contínua.

3.2. Requisitos do Sistema

Em linhas gerais o sistema deve permitir que qualquer pessoa possa se cadastrar e criar reuniões eletrônicas compostas por uma sala de bate-papo em tempo real e por uma ou mais votações. Uma reunião possui apenas uma sala de bate-papo, a qual oferece recursos básicos para a formatação das mensagens e identificação dos seus autores.

O usuário responsável pela criação da reunião, chamado de coordenador, pode criar e fechar votações sempre que necessário. O coordenador também determina quando o resultado de uma votação estará disponível aos participantes, o que pode ocorrer antes ou depois de seu encerramento pelo coordenador. O coordenador também pode convidar outros usuários para participar da reunião através de convites. Um usuário convidado poder aceitar ou não um convite para participar de uma reunião, operação que deve ficar registrada e ser encaminhada na forma de confirmação para o coordenador da reunião.

Ao aceitar um convite o usuário torna-se um participante credenciado para atuar em uma reunião. Sempre que um participante entra ou sai de uma reunião os demais participantes são notificados. Os usuários que não aceitaram um convite para uma determinada reunião uma vez, não deverão mais receber convites para aquela reunião, evitando práticas inadequadas de insistência no envio do convite.

O coordenador pode remover um participante de uma reunião sempre que desejar, desde que o participante não esteja atuando na reunião naquele momento. Os participantes que forem removidos podem ser convidados novamente para a reunião da qual foram removidos.

Os requisitos não funcionais desejáveis para o sistema compreendem diversos aspectos, muitos dos quais são contemplados pela própria plataforma J2EE [Sriganesh

et al. 2006] escolhida para a implementação do sistema, tais como: a atomicidade das operações dentro do sistema; a independência do banco de dados usado pela aplicação para a persistência das informações; e a utilização de formas, cores, ícones e elementos que mantenham a interação dos usuários com o sistema fácil e agradável. Outros requisitos importantes como portabilidade, escalabilidade e desempenho são intrínsecos à plataforma J2EE e conseqüentemente presentes no WebBEMS.

3.3. Aspectos de Projeto

Com base nos requisitos funcionais o sistema foi dividido em 5 componentes gerenciadores: *UserManager*, *InvitationManager*, *SessionManager*, *PoolManager* e *ChatManager*. A Figura 2 apresenta a arquitetura do WebBEMS utilizando diagrama de componentes de UML. Os estereótipos <<application>>, <<groupware>> e <<basic>> foram utilizados para representar componentes de aplicação, colaborativo e básico, respectivamente.

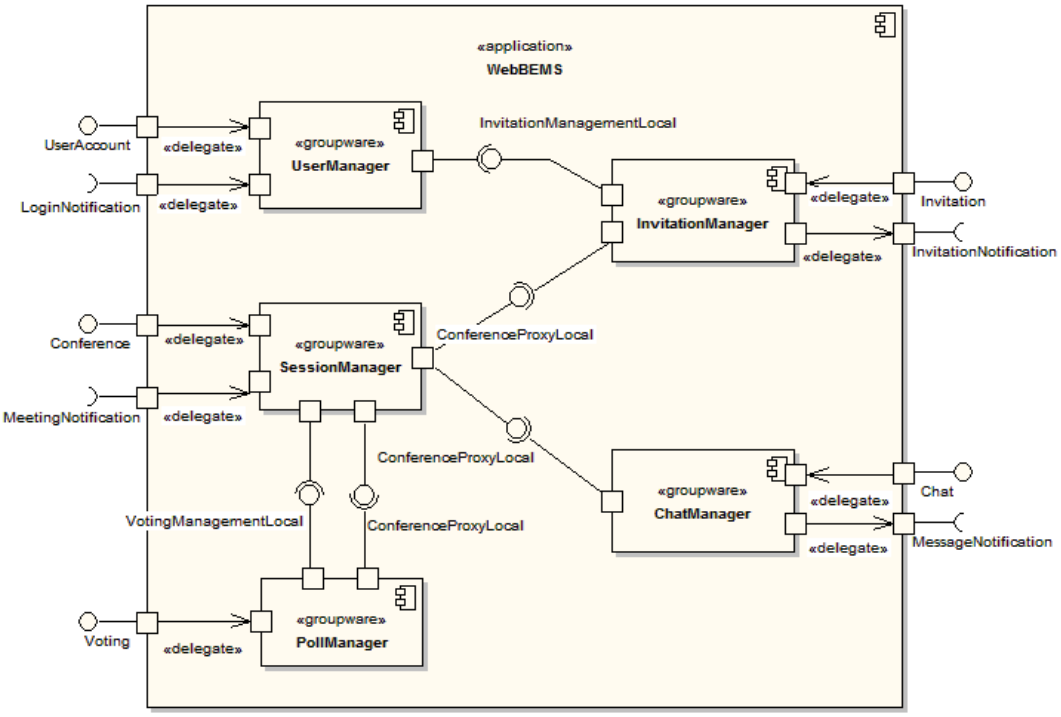

Figura 2. Arquitetura do WebBEMS

Cada um destes gerenciadores é composto por um conjunto de componentes interdependentes estruturados de acordo com o nosso modelo arquitetônico [de Farias *et al*. 2005]. Estes componentes implementam as camadas de usuário, colaboração e recursos deste modelo.

As camadas de interface dos componentes gerenciadores foram agrupadas e implementadas no componente *Interface* do WebBEMS (veja Figura 4). Este componente é composto por um conjunto de objetos que permitem acessar as funções dos demais componentes dentro do sistema. Cada operação solicitada pelo usuário é processada por um ou mais objetos e estes se encarregam de se comunicar com os

componentes do sistema. Dessa forma, o cliente só visualiza e interage com as telas propagadas pelo servidor.

Os componentes da camada de usuário com interfaces assíncronas foram modelados como componentes do tipo *Message Driven Bean* da plataforma J2EE. Os demais componentes da camada do usuário, juntamente com os componentes da camada de colaboração, foram modelados como componentes do tipo *Stateless Session Bean* da plataforma J2EE. Os componentes da camada de recursos foram modelados como componentes do tipo *Entity Bean* da plataforma J2EE.

A Figura 3 apresenta a estrutura interna do componente *InvitationManager*. O componente *InvitationNotifier* foi modelado como um *Message Driven Bean*. Os componentes *InvitationProxy* e *InvitationManagement* foram modelados como *Stateless Session Beans*. O componente *Invitation* foi modelado como *Entity Bean*.

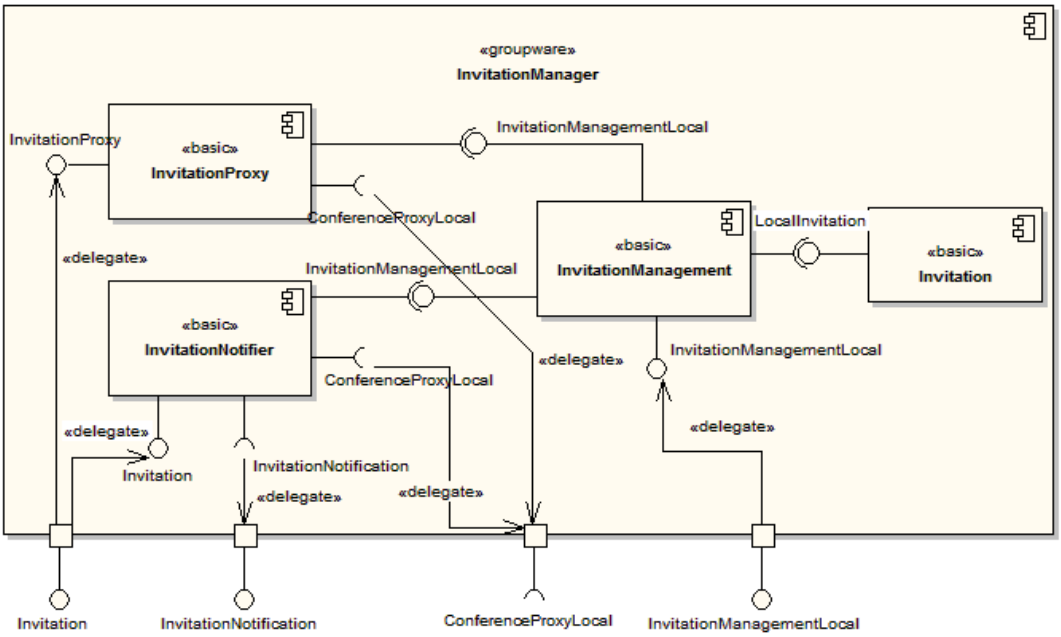

Figura 3. Componente InvitationManager

3.4. Aspectos de Interface e Comunicação

A interface Web foi desenhada para operar a partir das tecnologias J2EE *Servlet* e *JavaServer Pages* [Falkner e Jones 2003], em conjunto com o *framework Jakarta Struts* [Hightower 2004], que atribuem à aplicação outros padrões de projeto como *Model-View-Control*, *Front-Controller* e *Intercepting Filter*.

A natureza dinâmica das reuniões exige que os participantes sejam notificados sobre as ações dos demais participantes de uma reunião em tempo real (suporte à percepção). Para atender a este importante requisito a interface foi projetada de modo a tornar o WebBEMS capaz de operar como uma aplicação rica para Internet (*Rich Internet Application*). Neste sentido, a interface vai além de uma interface HTML padrão, limitada a executar requisições e processar respostas. Ela possui elementos ativos que a tornam mais rica e complexa.

O primeiro elemento presente na interface é um motor para execução de *Ashyncronous JavaScript And XML* (*AJAX*), que em linhas gerais é uma tecnologia que permite a execução de requisições HTTP assíncronas controladas programaticamente. Este recurso torna a interface capaz de enviar e receber informações ao sistema sem a necessidade de atualizar todo o conteúdo das páginas HTML carregadas pelo navegador Web, reduzindo o tempo de espera e enriquecendo a interação dos usuários.

O outro elemento aplicado na interface é um *Java Applet*, responsável por orquestrar o envio e recebimento de mensagens através da tecnologia *Java Message Service* (*JMS*). Este *Applet* não emprega nenhum tipo de recurso visual baseado em *frameworks* de janelamento *Java*, todas as interações entre o usuário e o *Applet* são feitas por meio de *links* e controles presentes na própria página HTML, graças ao uso da tecnologia *LiveConnect* suportada pelos principais navegadores.

Com estes dois elementos combinados, a interface central do WebBEMS é formada por uma única página HTML que embarca dentro de sua estrutura o *Applet*, formato conhecido como *Single Page Application* (*SPA*). Uma SPA é uma aplicação web executada completamente dentro de um navegador web.

Uma vez acessada a página, o que demanda um processo de autenticação, todos os conteúdos referentes aos convites, participantes, enquetes e mensagens de bate-papo são aplicados dinamicamente à página usando o motor *AJAX* sem a necessidade de mudar para outra página ou atualizar a página atual. A Figura 4 apresenta a representação completa dos principais elementos que compõe a aplicação e a infra-estrutura de comunicação utilizada pelos mesmos.

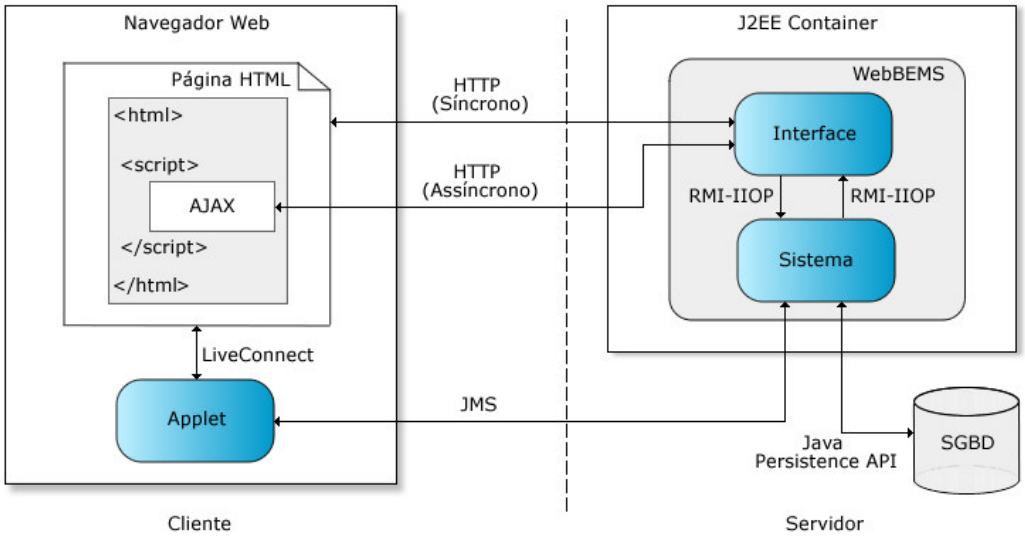

Figura 4. Arquitetura de comunicação do WebBEMS

4. Implementação do WebBEMS

4.1. Autenticação e autorização

O método de autenticação e autorização implementado no sistema baseia-se na propagação de credenciais do cliente para o servidor a cada requisição. O método de

autenticação e autorização baseada em papéis não foi utilizado porque no WebBEMS o papel de um usuário pode variar indefinidamente entre administrador e participante de uma reunião, o que inviabiliza a atribuição de papéis durante o processo de autenticação.

A autenticação é dividida em duas fases complementares:

- na primeira fase é feita a aquisição das credenciais do usuário através de um formulário de *login*. Estas credenciais são verificadas através de um método de autenticação do sistema, independente do *Java Authentication and Authorization Service* (*JAAS)*, e, caso sejam válidas, estas são registradas na sessão do usuário que é redirecionado para a área restrita da interface identificada por um URL diferenciado da área não restrita;
- dentro da área restrita, um filtro intercepta todas as requisições a fim de recuperar as credenciais do usuário presentes na sua respectiva sessão, para efetuar o processo de autenticação usando JAAS, que se encarrega de propagar as credenciais, já validadas pelo sistema.

A autorização usa as credenciais propagadas pela interface para confrontar as chamadas de um solicitante e verificar se o mesmo pode executar aquela operação. Por exemplo, um convite para uma reunião só pode ser enviado se o solicitante for o administrador da reunião em questão, do mesmo modo que um convite só pode ser confirmado pelo seu destinatário.

4.2. Transferência de mensagens

A transferência de mensagens (quer sejam mensagens de bate-papo ou convites) é outra parte importante do sistema e necessita de confiabilidade. No entanto a propagação de credenciais fornecidas pelo JAAS não funciona no sistema de mensagens J2EE, isto é, não é possível saber qual usuário enviou uma mensagem para o sistema e verificar se este possui autorização para executar a operação solicitada.

Uma das formas encontradas para contornar esta questão foi enviar a identificação do usuário dentro das mensagens. Para isto foi criada uma classe usada exclusivamente para que os usuários possam enviar mensagens para o sistema. Desta forma é possível encapsular os dados da mensagem juntamente com a identificação do usuário que a enviou.

Esta estratégia resolve a troca de mensagens que partem dos clientes para o WebBEMS. Contudo a troca de mensagens no sentido inverso deve ser tratada de outra forma. Além de identificar o destinatário da mensagem é preciso que uma mensagem seja consumida apenas pelo seu destinatário e ninguém mais. Para atender a este requisito cada cliente cria uma fila de mensagens temporária. Esta fila temporária é provida pelo JMS e funciona da mesma forma que uma fila convencional, a diferença é que apenas o criador a fila pode ler as mensagens enviadas para a mesma.

Deste modo sempre que o *Applet* é iniciado ele cria uma fila temporária e registra a mesma no WebBEMS para que este saiba qual a fila que cada usuário está usando para receber suas mensagens vindas do sistema. Antes de enviar uma mensagem para um usuário o sistema recupera a fila que está sendo usada pelo usuário e envia a mensagem para a mesma.

4.3. Interface web

A interface possui suporte a múltiplos idiomas e é totalmente baseada em HTML. Ela é composta por quadros projetados para se assemelharem com pequenas janelas, através das quais os usuários invocam os serviços do WebBEMS. Ao todo são seis quadros, dispostos simultaneamente na tela, representando as reuniões, os participantes, as enquetes, a sala de bate-papo, os convites recebidos pelo usuário e confirmações de convites enviados pelo usuário. A interface permite que o usuário maximize e minimize o conteúdo de cada quadro, de acordo com sua necessidade. Todos os formulários para cadastro, edição e mensagens são embutidos dentro dos quadros dando-lhes um aspecto de interfaces independentes semelhantes a *Portlets*.

A Figura 5 apresenta uma janela contendo a interface com todos os quadros maximizados e com os respectivos conteúdos a disposição do usuário, que permitem que este tenha percepção das interações dos demais usuários. Caso o usuário deseje se concentrar em apenas determinados quadros ele pode minimizar os quadros que não são de interesse no momento. Por exemplo, durante o cadastro de uma nova votação é possível manter apenas alguns quadros maximizados evitando com isso possíveis distrações. A Figura 6 ilustra está situação. Nesta figura, os quadros de convites e confirmações estão minimizados.

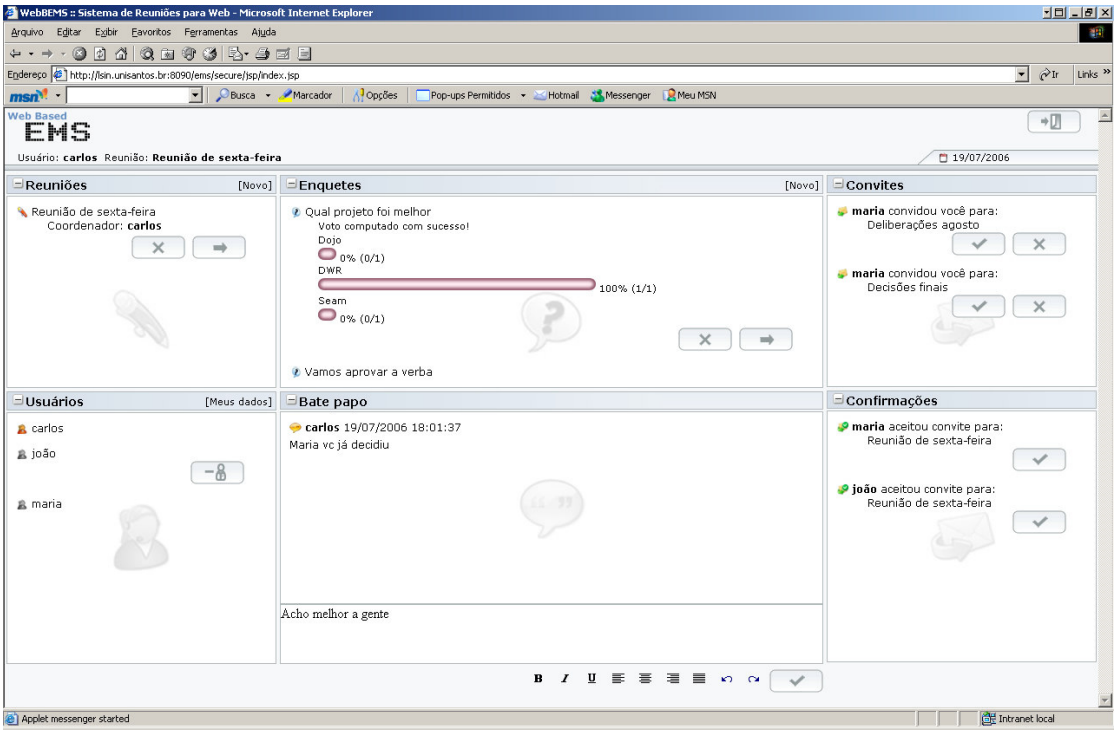

Figura 5. Interface do WebBEMS

Para acessar o conteúdo dinâmico dos quadros e submeter os dados para o servidor a interface utiliza a tecnologia *AJAX*. O motor usado na interface, chamado *Clean AJAX Engine*, simplifica significativamente o processo de construção dos objetos *JavaScript* necessários para a utilização desta tecnologia em diferentes navegadores.

Alguns quadros utilizam o serviço de mensagens *JMS* para acessar e enviar dados para o servidor. Estes quadros utilizam a tecnologia *LiveConnect* para receber e enviar dados ao *Applet* embarcado na página que possui acesso ao serviço *JMS*. O *Applet* possui assinatura via certificado digital para permitir a habilitação de recursos normalmente inacessíveis para *Applets* executados dentro de navegadores.

As informações acessadas através do motor *AJAX* trafegam entre o servidor e o cliente já no formato HTML, sem a necessidade de nenhuma transformação no momento da exibição das informações no cliente. Já as informações acessadas pelo *Applet* trafegam entre o servidor e o cliente na forma de objetos *Java*. A exibição destas informações utiliza *templates* HTML disponibilizados com o *Applet*, onde as informações são inseridas para que possam ser exibidas adequadamente.

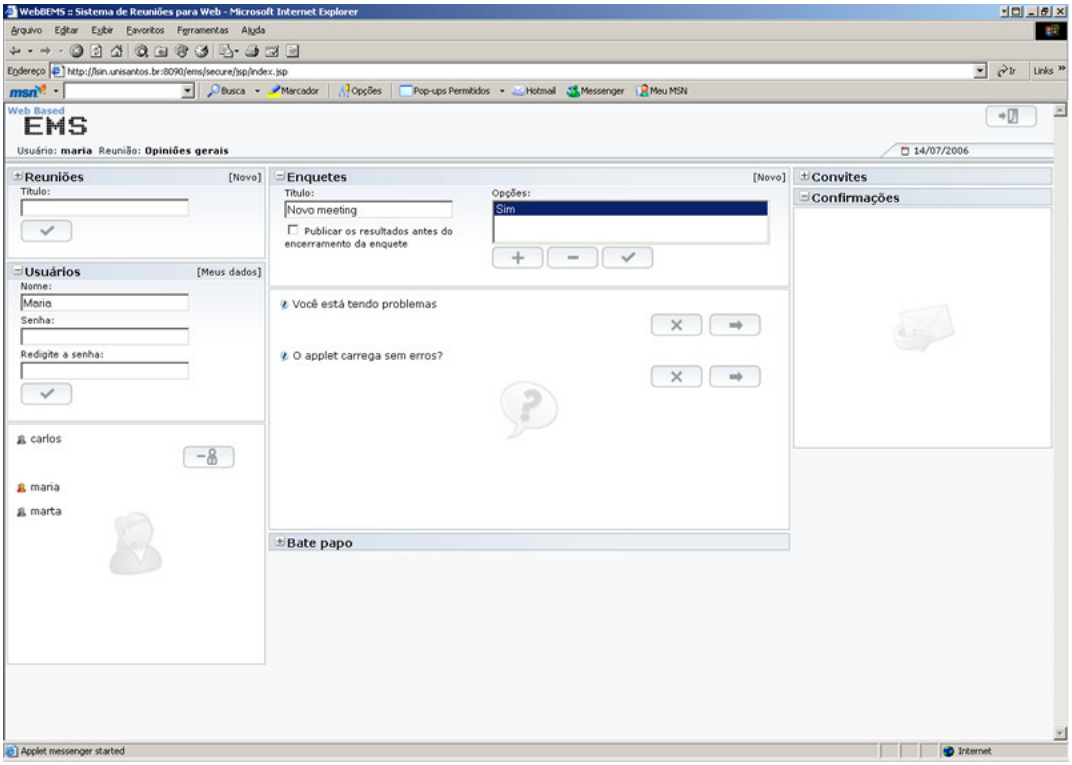

Figura 6. Interface para o cadastro de enquete

4.4. Testes

Os testes de unidade concentraram-se nos componentes EJB e foram executados utilizando a prática de testes *in-container*, a partir do *framework Cactus*, na qual os testes de unidade são executados dentro do ambiente provido pelo EJB *container*. Os testes de aceitação foram feitos a partir da simulação de reuniões observando o comportamento e performance da aplicação como um todo, isto é, observando os recursos providos tanto pelo lado servidor quanto pelo lado cliente.

5. Trabalhos Relacionados

Sistemas de reunião eletrônica são sistemas colaborativos concebidos para prover suporte eletrônico a diferentes tipos de reuniões, desde reuniões síncronas face-a-face até reuniões assíncronas distribuídas [Nunamaker *et al*. 1991]. Em geral estes sistemas têm por objetivo aumentar a produtividade e satisfação dos participantes envolvidos nestas atividades. Tais sistemas podem, por exemplo, dar suporte ao aprendizado colaborativo no contexto educacional ou dar suporte ao trabalho cooperativo e a tomada de decisão de maneira geral.

No contexto da educação, enfatiza-se o valor dos sistemas de suporte a grupos em geral, definindo-o como um sistema computacional que permite que um grupo de pessoas tanto possa entrar com dados como também compartilhá-los [Weatherall 2000]. Este tipo de sistema tem sido usado para melhorar a efetividade de reuniões face-a-face, onde a discussão é suplementada pela possibilidade dos participantes colocarem fatos, opiniões, questões, votos, etc. Nesta situação, geralmente cada participante da reunião tem seu próprio computador. Já no cenário de reunião eletrônica, um sistema de suporte a colaboração permite que os participantes possam participar de diferentes atividades a partir do trabalho, de casa, etc., o que é particularmente relevante no contexto de um treinamento.

Em [de Farias *et al*. 2005, de Farias *et al*. 2006] mostrou-se a relevância de um modelo de arquitetura baseada em componentes para os sistemas e plataformas de aprendizado eletrônico. A aplicação de um modelo de componentes projetado especificamente para atender sistemas colaborativos foi ilustrada através da implementação de um sistema de votação eletrônica baseado na Web, denominado Electronic Voting System (EVS). O EVS basicamente permite que os seus usuários criem uma enquete e/ou votem nela. No aprendizado eletrônico o sistema permite que os estudantes incluam anonimamente idéias e preferências através da criação de uma enquete, votem secretamente e verifiquem a existência de consenso ou não no grupo sobre uma determinada questão, apontando os pontos que devem ser discutidos ou debatidos [Yankelovich *et al*. 2004].

O WebBEMS é um sistema de reunião eletrônica que foi desenvolvido com o objetivo de contemplar os principais recursos que permitem dar suporte e ampliar a colaboração entre os usuários. O sistema pode ser utilizado em diferentes contextos mas especialmente no aprendizado eletrônico permite a utilização de abordagens pedagógicas onde a discussão e a busca por uma posição de consenso ou de maioria sobre uma determinada questão é necessária. O WebBEMS combina as funcionalidades de um sistema de votação eletrônica (i.e., o EVS) com as funcionalidades básicas de uma ferramenta de bate-papo para dar suporte a este tipo de atividade. Como apontado por [Kennedy and Cutts 2005], a adição do componente de bate-papo dá a oportunidade aos participantes de uma atividade colaborativa, especialmente no contexto educacional, de discutirem as questões e suas respostas uns com os outros, associando a votação com uma discussão com os seus pares.

6. Conclusão

Este artigo apresentou o projeto e o desenvolvimento do WebBEMS, um sistema baseado em componentes para reuniões eletrônicas pela Internet. Toda a especificação e

desenvolvimento do WebBEMS foi baseado em um ciclo de desenvolvimento incremental e iterativo, que conferiu grande dinâmica e agilidade ao projeto.

O projeto do WebBEMS foi norteado por um modelo arquitetônico que têm sido proposto para o desenvolvimento de sistemas colaborativos baseados em componentes. A adoção deste modelo facilitou a identificação e estruturação dos componentes ao longo do processo de especificação e implementação do sistema. Adicionalmente, utilizamos diversos padrões de projeto de modo a oferecer uma grande flexibilidade de manutenção e reengenharia ao sistema. A interface com usuário do WebBEMS possui um design atual voltado a aplicações Web 2.0 e que promove uma maior interatividade entre os usuários se comparada com as interfaces Web tradicionais.

No que se refere ao reuso dos componentes em comum entre o EVS [de Farias *et al*. 2005, de Farias *et al*. 2006] e o WebBEMS, que são os componentes *UserManager* e *PollManager*, considera-se que o nível de reutilização foi baixo. Isto se deve ao fato de que ocorreram alterações significativas nestes componentes com o acréscimo de novas funcionalidades. No componente *UserManager* o EJB responsável por controlar os usuários logados no sistema havia sido implementado no EVS com base no padrão *Singleton*. Contudo o padrão *Singleton* é um padrão desaconselhado para a os componentes EJB, e isto motivou a reconstrução de parte do *UserManager*. No componente *PollManager*, a inclusão de novos requisitos funcionais ampliou significativamente o tamanho do componente, o que motivou o seu desdobramento em 3 componentes no WebBEMS (*PollManager*, *SessionManager*, e *ParticipantManager*).

Em princípio, pretende-se utilizar o WebBEMS no aprendizado eletrônico, com suporte a interações presenciais e à distância. Adicionalmente, planeja-se ampliar as funcionalidades do sistema através da adição de serviços de calendário em grupo e filtro de mensagens.

Agradecimentos

Trabalho realizado com o auxílio da FAPESP sob o projeto número 2003/08279-2.

Referências

Bass, L., Clements, P. and Kazman, R. (1997). *Software Architecture in Practice*. Addison-Wesley.

Brusilovsky, P. (2004). KnowledgeTree: A Distributed Architecture for Adaptive E-Learning. *Proceedings of the 13th international World Wide Web Conference (WWW´04)*, pp. 104-113.

de Farias, C. R.G. (2002). *Architectural Design of Groupware Systems: a Component-Based Approach*. PhD Thesis, University of Twente, Enschede, the Netherlands.

de Farias, C.R.G., Gonçalves, C.E., Rosatelli, M.C., Ferreira Pires, L. and van Sinderen, M. (2005). An Architectural Model for Component Groupware. *Proceedings of the 11th Int. Workshop on Groupware (CRIWG´05)*, Lecture Notes in Computer Science, 3706, Springer-Verlag, pp. 105-120.

de Farias, C.R.G., Rosatelli, M.C., Gonçalves, C.E. (2006). Applying a Component-Based Architectural Model in the Development of e-Learning Systems. *Proceedings*

of the 6th IEEE International Conference on Advanced Learning Technologies (ICALT´06), IEEE Computer Society Press, pp. 247-251.

Falkner, J., Jones, K. (2003). *Servlets and JavaServer Pages: The J2EE Technology Web Tier*. Addison-Wesley.

Fuks, H., Raposo, A. B., Gerosa, M. A. and Lucena, C. J. P. (2005). Applying the 3C Model to Groupware Development. In *International Journal of Cooperative Information Systems (IJCIS)*, 14(2-3), pp. 299-328

Hightower, R. (2004). *Jakarta Struts Live*. SourceBeat.

Jeffries, R., Hendrickson, C., Anderson, A. (2000). *Extreme Programming Installed*. Addison-Wesley.

Kennedy, G. E. and Cutts, Q. I. (2005). The Association Between Students' Use of an Electronic Voting System and Their Learning Outcomes. In *Journal of Computer Assisted Learning*, 21(4), Pp. 260-268.

Nunamaker, J.F., Dennis, A.R., Valacich, J.S., Vogel, D.R. and George, J.F. (1991). Electronic meeting systems to support group work. *Communications of the ACM*, 34 (7), pp. 40-61.

OMG (2007a): *Unified Modeling Language: Infrastructure*. OMG Specification, version 2.1.1, Object Management Group.

OMG (2007b): *Unified Modeling Language: Superstructure*. OMG Specification, version 2.1.1, Object Management Group.

Pukkhem, N., Vatanawood, W. (2005). Instructional Design Using Component-Based Development and Learning Object Classification. *Proceedings of the Fifth IEEE International Conference on Advanced Learning Technologies (ICALT'05)*, p. 492-494.

Sriganesh, R. P., Brose, G. and Silverman, M. (2006). *Mastering Enterprise JavaBeans 3.0*. Wiley Publising Inc.

ter Hofte, G. H. (1998): Working Apart Together: Foundations for Component Groupware. PhD Thesis, Telematics Institute, the Netherlands.

van Vliet, H. (2000). Software Engineering: Principles and Practice. John Wiley & Sons, USA.

Weatherall, A. (2000). Improving and Focusing a Training Course using GroupSystems Electronic Meetings. Proceedings of the 33rd Hawaii International Conference on System Sciences, IEEE Computer Society Press, pp. 292- 300.

Yankelovich, N., Walker, W., Roberts, P., Wessler, M., Kaplan, J., and Provino J. (2004). Meeting central: Making Distributed Meetings More Effective. In *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work (CSCW´04)*, ACM Press, pp. 419-428.