Marcelo Andrade da Gama Malcher

Um Middleware e Aplicativo para Apresentação Colaborativa para Dispositivos Móveis

DISSERTAÇÃO DE MESTRADO

DEPARTAMENTO DE INFORMÁTICA

Programa de Pós-Graduação em Informática

Rio de Janeiro, agosto de 2007

PONTIFÍCIA UNIVERSIDADE CATÓLICA
DO RIO DE JANEIRO

Marcelo Andrade da Gama Malcher

Um Middleware e Aplicativo para Apresentação Colaborativa para Dispositivos Móveis

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do título de Mestre pelo Programa de Pós-Graduação em Informática da PUC-Rio.

Orientador: Prof. Markus Endler

Rio de Janeiro

Agosto de 2007

PONTIFÍCIA UNIVERSIDADE CATÓLICA
DO RIO DE JANEIRO

Marcelo Andrade da Gama Malcher

Um Middleware e Aplicativo para Apresentação Colaborativa para Dispositivos Móveis

Dissertação apresentada como requisito parcial para obtenção do título de Mestre pelo Programa de Pós-Graduação em Informática do Centro Técnico e Científico da PUC-Rio.

Prof. Markus Endler
Orientador
Departamento de Informática - PUC-Rio

Prof. Renato Fontoura Cerqueira
Departamento de Informática - PUC-Rio

Prof. Alberto Raposo
Departamento de Informática - PUC-Rio

Prof.ª Renata Mendes de Araújo
Departamento de Informática Aplicada - UNIRIO

Prof. José Eugenio Leal
Coordenador Setorial do Centro Técnico Científico – PUC-Rio

Rio de Janeiro, 24 de agosto de 2007

Marcelo Andrade da Gama Malcher

Graduado em Ciência da Computação pela Universidade Federal do Pará (UFPA) em 2005. Atualmente, integra o grupo de pesquisadores do LAC (Laboratory of Advanced Collaboration) da PUC-Rio, desenvolvendo pesquisa na área de Sistemas Distribuídos.

Ficha Catalográfica

Malcher, Marcelo Andrade da Gama

Um middleware e aplicativo para apresentação colaborativa para dispositivos móveis / Marcelo Andrade da Gama Malcher; orientador: Markus Endler. – 2007.

109 f. : il. (col.) ; 30 cm

Dissertação (Mestrado em Informática)– Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2007.

Inclui bibliografia

1. Informática – Teses. 2. Compartilhamento de apresentações. 3. Colaboração síncrona. 4. Ensino interativo. 5. Sistemas distribuídos. 6. Computação móvel. 7. Aplicações sensíveis a contexto. I. Endler, Markus. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Este trabalho é dedicado aos meus pais, Luiz Paulo e Beth, que sempre me deram muito amor, carinho e força em todos os momentos de minha vida.

E aos meus irmãos, Fabrício e Marina, por toda amizade e apoio durante o desenvolvimento deste trabalho.

Agradecimentos

Primeiramente agradeço ao meu orientador, Professor Markus Endler, por toda a paciência e compreensão, sempre demonstrada durante as aulas e reuniões, e a confiança depositada em mim para a conclusão deste trabalho.

Agradeço ao meu pai, Luiz Paulo, por todas as palavras de incentivo, toda a preocupação e paciência despendida comigo. À minha mãe, Beth, por todo o amor e carinho, por todas as conversas e conselhos dados durante o mestrado. Aos meus irmãos pelo apoio e companheirismo. Também não poderia esquecer a minha tia Selma, que me recebeu como um filho desde minha chegada no Rio.

Agradeço também todos os membros do LAC (*Laboratory for Advanced Collaboration*) pela amizade e companheirismo. Agradeço ao Viterbo, sempre um amigo para conversar e pedir conselhos, ao Gustavo, companheiro de trabalho nos feriados e finais de semana, e em especial ao Ricardo, por sempre ter me ajudado com paciência e atenção a resolver os diversos problemas que surgiram durante o desenvolvimento deste trabalho.

A todas as minhas tias, primos e amigos de Belém e do Rio de Janeiro por todo carinho e amizade. Em especial os primos Tavinho e Ilana, e os amigos Bernar, Hugo, Rodrigo, Renato, Laiola, Marcelinho, Suzana e Sônia.

Aos membros da banca pelos comentários pertinentes e pelas revisões precisas. Agradeço também aos professores e funcionários do Departamento de Informática da PUC-Rio que colaboraram para a conclusão deste trabalho.

Por último, o meu agradecimento mais especial a Deus, que me deu a oportunidade de realizar este trabalho, no qual pude fazer novas amizades. Agradeço a Ele também pela vivência e aprendizado adquirido durante estes anos de mestrado, que com certeza serão muito importantes para o meu futuro profissional.

Resumo

Malcher, Marcelo Andrade da G., Endler, Markus. **Um Middleware e Aplicativo para Apresentação Colaborativa para Dispositivos Móveis.** Rio de Janeiro, 2007. 109p. Dissertação de Mestrado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

A atual evolução dos dispositivos computacionais móveis e a crescente ubiqüidade de redes *sem fio* possibilitam o desenvolvimento de serviços e aplicativos para colaboração entre usuários móveis nos mais variados ambientes como em domicílios, lugares públicos, universidades, empresas, entre outros. Em uma sala de aula, acredita-se que o uso de dispositivos móveis (com capacidade de comunicação sem fio) torna o aprendizado mais interativo e estimulante. Este trabalho descreve um aplicativo distribuído, denominado iPH (*Interactive Presenter for Handhelds*), que possibilita o compartilhamento e a co-edição de transparências entre o instrutor e os aprendizes em sala de aula, e os componentes de *middleware* utilizados no desenvolvimento do mesmo. O iPH pode ser executado em diferentes tipos de dispositivos como *tablet pcs*, *notebooks* e *handhelds (palmtops* ou *smartphones)*, e acessa informações de contexto computacional do dispositivo para efetuar auto-adaptações na sua funcionalidade, para entre outros, melhorar a interação com o usuário.

Palavras-chave

Compartilhamento de Apresentações; Colaboração Síncrona; Ensino Interativo; Sistemas Distribuídos; Computação Móvel; Aplicações sensíveis a contexto.

Abstract

Malcher, Marcelo Andrade da G., Endler, Markus. **A Middleware and an Application for Collaborative Presentation Sharing on Handhelds.** Rio de Janeiro, 2007. 109p. Master Thesis - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

The ongoing improvement of portable devices and the increasing ubiquity of wireless networks enable the development of services and applications for *any-place-any-time* collaboration among mobile users in many different environments, such as at home, in public areas, in universities, in companies, among others. It is expected that the use of portable, wireless-enabled devices in classrooms improves the interaction and engagement in the learning process. This work describes a distributed application named iPH (Interactive Presenter for Handhelds) that supports the sharing and co-edition of presentations among an instructor and students of a classroom, as well as the middleware components used for the development of iPH. This system can be executed on a wide range of devices, such as tablets, notebooks and handhelds (palmtops or smartphones), and uses the device's context information to adapt itself to improve, for example, the interaction with the user.

Key words

Presentation Sharing; Synchronous Collaboration; Active Learning; Distributed Systems; Mobile Computing; Context-aware applications.

Sumário

Lista de figuras

Lista de tabelas

1
Introdução

Dispositivos computacionais móveis estão se tornando cada vez mais leves, com maior capacidade de processamento e de armazenamento, e com preços mais acessíveis. Ao mesmo tempo a difusão das redes *wireless* faz com que em muitos ambientes, como domicílios, lugares públicos, *shopping centers*, universidades e empresas, seja possível utilizar esses dispositivos móveis para acessar a Internet, compartilhar arquivos, trocar mensagens instantâneas com outras pessoas, entre outras formas de interação.

O uso cada vez mais difundido desses dispositivos faz com que surjam novas possibilidades de colaboração, seguindo a tendência de uma computação ubíqua [Prekop, 2003]. Aproveitando-se os recursos de comunicação sem fio (802.11b/g, *bluetooth*, infravermelho) disponíveis em um ambiente, é possível desenvolver serviços ou aplicativos para colaboração entre dispositivos computacionais.

Este tipo de ambiente, capaz de suportar comunicação móvel e colaboração entre diferentes dispositivos será chamado de **ambiente colaborativo móvel.** Como ilustrado na Figura 1, em um ambiente colaborativo móvel, vários dispositivos computacionais estão presentes, sejam eles fixos ou móveis, como celulares, *palmtops* ou *notebooks*. Dada a presença de uma forma de comunicação móvel, por exemplo, uma rede sem fio 802.11b, esses dispositivos podem colaborar entre si, através de aplicações especialmente desenvolvidas para trabalho colaborativo.

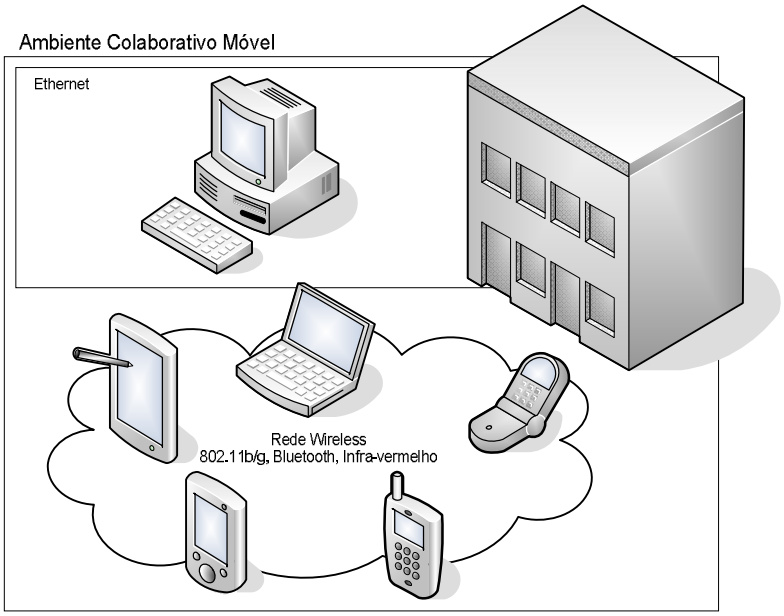

Figura 1 - Um ambiente colaborativo móvel

Devido à heterogeneidade dos dispositivos que podem estar presentes em um ambiente, o modo como esses dispositivos se comunicam e colaboram entre si pode ser complexo. Em um único ambiente podem existir dispositivos com diferentes características em termos da capacidade de processamento, de memória, do tamanho de *display*, de acesso à rede, etc..

A própria mobilidade dos usuários participantes de uma colaboração, com constantes mudanças de localização e de conectividade à rede, bem como, a disponibilidade variável de recursos computacionais de cada dispositivo (i.e. nível de energia restante), requer o acesso às informações de contexto para maximizar as possibilidades de interação e acesso à informação nas aplicações desenvolvidas para este tipo de ambiente. Segundo [Dey, 2000], uma aplicação deve utilizar informações de contexto para se adaptar às condições correntes e assim prover uma melhor interação com o usuário.

Ao desenvolver aplicações que implementem formas de colaboração entre diferentes tipos de dispositivos, os desenvolvedores precisam analisar diversas questões: quais são os tipos de dispositivos nos quais estas aplicações deverão executar, como são os grupos de colaboração (pré-definidos, dinâmicos, etc.), quais são os papéis de seus membros, qual é o grau de sincronização entre os estados dos dispositivos necessário para a forma de colaboração a ser implementada, como deverá ser a interação usuário-dispositivo, quais são as

informações de contexto relevantes para facilitar o uso da aplicação e melhorar a interatividade entre os usuários, e de cada usuário e seu dispositivo.

Vários trabalhos apontam para os benefícios do uso de dispositivos móveis em sala de aula, para tornar as aulas mais interativas e estimulantes [Nilson, 2005]. No entanto, em um cenário típico como uma sala de aula, um aluno utilizando um *tablet pc* tem muito mais facilidade para interagir com o professor e os demais alunos do que um aluno que está utilizando um celular, com um teclado limitado, para a entrada de dados, e um *display* de tamanho reduzido. Além disso, um aluno com *tablet pc* terá uma melhor visualização de conteúdos sendo apresentados. Contudo, seria desejável que aplicações colaborativas de apoio ao ensino executassem tanto em dispositivos mais poderosos como *tablet pcs*, como também em celulares ou *palmtops*, dado que estes últimos, além de serem mais baratos e serem mais difundidos, também são mais adequados ao uso no dia-a-dia, por serem menores e mais leves.

Em especial, uma aplicação que possibilite o compartilhamento de apresentações entre os participantes de uma aula, e que ofereça a estes participantes a possibilidade de dar e compartilhar contribuições digitais, é uma ferramenta que, se utilizada adequadamente, pode aumentar a interatividade em sala de aula, e assim aumentar a eficiência do processo de aprendizado [Maite, 2003].

Como mencionado, o ideal é que essa aplicação possa ser executada em diferentes tipos de dispositivos, como *notebooks*, *tablet pcs* e *handhelds*. No entanto, para tal, as seguintes questões relativas a esta aplicação devem ser tratadas pelos desenvolvedores, a fim de prover uma melhor funcionalidade e uma melhor interação da aplicação com os usuários:

- Como o conteúdo de uma apresentação será mostrado em *handhelds*? Pois a dimensão da tela destes dispositivos é menor. Portanto, isto requer uma adaptação de conteúdo;
- Como será a interação homem-máquina para cada um dos diferentes dispositivos? Em especial, precisa-se determinar como os participantes poderão co-editar os quadros[1], quais formas de edição

[1] Ao mencionar quadro de uma apresentação, está se referindo tanto a uma área livre para edição (*whiteboard*) como para um dos *slides* de uma apresentação.

serão habilitadas, e como visualizar as diferentes contribuições nos dispositivos, entre outras;

- Questões de sincronização da visualização da apresentação: quem controlará o avanço e o retrocesso de quadros da apresentação? Quando os dispositivos deverão ser sincronizados? Existirá um participante que coordenará a apresentação? Outras questões devem ser resolvidas como quando os participantes devem visualizar determinado quadro, quando um participante poderá enviar sua contribuição aos outros participantes, quem receberá estas contribuições, entre outras;
- Como informações a respeito do ambiente colaborativo móvel, como por exemplo, a localização dos participantes ou o estado atual dos recursos de seus dispositivos deve ser usado para melhorar a interação entre os participantes? Por exemplo, dispositivos de colaboradores com pouca memória disponível devem descartar quadros da apresentação já visualizados há muito tempo a fim de poder receber novos quadros criados pelo mestre. Ou então, poder-se-ia usar a informação sobre a localização geográfica do dispositivo. Neste caso, por exemplo, somente usuários localizados dentro (ou nas proximidades) da sala de aula poderiam enviar contribuições para o mestre. Assim, seria possível garantir que as contribuições sejam aderentes às questões levantadas pelos participantes presenciais.

Este tipo de apresentação compartilhada, quando utilizada para ensino interativo, onde participantes de uma sessão de colaboração podem interagir utilizando seus dispositivos móveis, será chamado neste documento de **apresentação colaborativa.** O objetivo de uma apresentação colaborativa é melhorar o engajamento e a atenção dos participantes ao conteúdo lecionado, e incentivar o debate sobre esse conteúdo.

1.1. Motivação do Trabalho

Existem alguns sistemas para **apresentação colaborativa**. Em geral, estes aplicativos (ou serviços) oferecem aos usuários apenas o compartilhamento e a co-edição de apresentações visualizadas por participantes de uma sessão, mas apresentam algumas limitações tais como: a execução somente em um determinado tipo de dispositivo (geralmente *notebook* e *tablet pc*); falta suporte à apresentações; e arquitetura centrada em servidor, o que causa problemas de escalabilidade e ponto central de falha no servidor; entre outras.

Existem também na literatura os mais diversos tipos de aplicações sensíveis ao contexto (*context-awareness*) [Dey, 2000]. Essas aplicações possuem a capacidade de se adaptar a algumas características do ambiente computacional e físico do usuário e fazem isso através do acesso a sistemas provedores de contexto. Com isso, podem buscar um melhor desempenho em diferentes situações de execução da aplicação (e.g. baixa/alta utilização da CPU, boa/má qualidade do enlace sem fio, nível alto/baixo de energia da bateria, etc.) e uma melhor adequação de sua funcionalidade e da interatividade entre o usuário, o seu dispositivo computacional, e o meio em que se encontra [Coutaz, 2005].

A inexistência de uma aplicação que possua as características mencionadas acima, a saber, o funcionamento tanto em dispositivos de mão - *handhelds* (*smartphones* e *palmtops)* como em *desktops* e *tablet pcs*, o suporte a apresentações, o compartilhamento e a co-edição de apresentações e a adaptabilidade baseada em contexto, motivou o presente trabalho.

1.2. Objetivo

Este trabalho tem como objetivo desenvolver um *middleware* e uma aplicação que permita o compartilhamento e a co-edição simultânea de uma apresentação colaborativa. Este *middleware* e aplicação devem executar em diferentes tipos de dispositivos, como computadores pessoais, *notebooks*, *tablet pcs*, *palmtops* e *smartphones* interconectados através de uma rede local sem fio *wi-fi* (IEEE 802.11). Como objetivo adicional, o aplicativo deverá, para fins experimentais, acessar informações de contexto computacional, e sofrer

adaptações de acordo com estas informações, caracterizando-se assim, um aplicativo sensível a contexto.

A aplicação adotará o modelo de colaboração do Classroom Presenter [Anderson, 2004a], com algumas modificações. Segundo este modelo, um dos participantes deverá criar uma **sessão de colaboração**, na qual os demais participantes deverão ingressar a fim de compartilhar a apresentação. O participante que criar a sessão fará o papel de **mestre**, controlando o andamento da apresentação, e decidindo qual quadro deve ser visualizado por todos os participantes. A aplicação deve permitir que as contribuições de cada um dos participantes da sessão de colaboração possam ser visualizadas (em cima de quadros da apresentação original) pelos demais participantes, de acordo com o interesse do mestre.

Os participantes deverão ser capazes de criar suas contribuições de dois modos: usando traços digitais (*digital ink*[2]) e texto. Com a *ink*, um participante poderá desenhar sobre um quadro de uma apresentação. Com o texto, um participante deverá escolher o local (no quadro) onde deseja inserir o texto, a sua fonte e cor. Ambos os tipos de contribuição poderão ser enviados para o mestre, que poderá exibir para todos os demais participantes as contribuições recebidas de cada participante. Também deverão existir formas de apagar as contribuições feitas, sejam elas desenhos ou textos.

A aplicação também poderá ser executada quando estiver desconectada de uma rede móvel. Todas as funções, exceto aquelas que estão relacionadas à sessão de colaboração, como o recebimento de quadros, a sincronização e o envio de contribuição estarão desabilitadas. Quanto o participante estiver conectado, e este perder sua conexão, o participante poderá continuar a fazer suas contribuições localmente até recuperar sua conexão, quando então ele poderá enviar ao mestre todas as contribuições feitas no período desconectado, além de visualizar todos os demais quadros da apresentação que esteja sendo compartilhada por todos. Além disso, o mestre poderá atualizar o participante com todas as informações que este não recebeu quando estava desconectado. Assim, o estado de sua apresentação será sincronizado com o estado atual da apresentação do mestre.

---

[2] O termo *ink* quando mencionado neste documento se refere tanto a caneta utilizada comumente por usuários de *palmtops* e *tablet pc* para interagir com as interfaces gráficas em substituição ao *mouse*, como também aos próprios desenhos realizados utilizando esta caneta.

Deverão ser criadas também formas de acesso às informações de contexto computacional e formas de utilizar estas informações para adaptação da aplicação em determinadas ocasiões durante a sessão.

1.3. Estrutura da Dissertação

Esta dissertação está organizada como a seguir. No Capítulo 2 são apresentadas e analisadas as aplicações de colaboração existentes, que apresentam características semelhantes às do trabalho proposto.

No Capítulo 3 são apresentados os conceitos que fundamentam esse trabalho, os principais requisitos da aplicação e motivações para o uso de informações de contexto pela ferramenta.

No Capítulo 4 são abordados os *middlewares* e os componentes utilizados e desenvolvidos para tratar da comunicação e sincronização entre os dispositivos, e o acesso às informações de contexto.

No Capítulo 5 descreve-se a aplicação para apresentação colaborativa desenvolvida, destacando-se seus conceitos e funcionalidades, e apresenta um cenário de uso da aplicação.

O Capítulo 6 descreve os detalhes de implementação da aplicação desenvolvida.

O Capítulo 7 trata da realização de testes para avaliar a viabilidade do uso da ferramenta em dispositivos móveis.

Por fim, no Capítulo 8 são apresentadas as considerações finais sobre o trabalho, destacando as principais contribuições desta dissertação e possíveis trabalhos futuros.

2
Trabalhos Relacionados

Com relação a compartilhamento simultâneo e co-edição de apresentações, existem diversos trabalhos relacionados publicados na literatura. No entanto, a forma como a colaboração é realizada varia entre os trabalhos. Por exemplo, enquanto alguns fazem uso do paradigma de uma sala de aula, onde um participante coordena o andamento da apresentação fazendo o papel do professor, em outros trabalhos a forma de colaboração é menos rígida, e mais descentralizada, uma vez que são voltados mais para discussões e menos para apresentações controladas por um único participante. Esses trabalhos serão detalhados neste capítulo.

Com relação a aplicações sensíveis a contexto, existe uma grande quantidade e variedade de aplicações, porém não encontramos nenhuma diretamente relacionada ao compartilhamento e à co-edição de apresentações. Logo, não será apresentado nenhum trabalho relacionado a esta característica da aplicação proposta.

2.1.
Classroom Presenter

O Classroom Presenter (CP) [ClassroomPresenter, 2007] foi desenvolvido pela Universidade de Washington e é o aplicativo que inspirou o desenvolvimento da aplicação desenvolvida nesse trabalho. O CP foi concebido para uso em *tablet pcs* e tem como objetivo facilitar o ensino em uma sala de aula disponibilizando uma interface em que todos os participantes visualizam em seus dispositivos o mesmo que o participante controlador (mestre) da apresentação.

Todas as anotações realizadas pelo mestre são instantaneamente visualizadas pelos outros participantes. Essas anotações podem ser realizadas de dois modos: utilizando o modo caneta ou o modo marca-texto, ambos com a opção de diferentes cores. Os outros participantes também podem editar um quadro da apresentação a partir de seus *tablets pc* e enviar esta colaboração para o

mestre, que poderá decidir se irá disponibilizá-la para todos os demais participantes [Anderson 2006].

Um cenário típico de uso do CP é uma aula em que os alunos são convocados a solucionar um problema enunciado pelo instrutor. Os alunos podem então enviar as suas respostas ao professor, que pode analisá-las, e escolher uma ou mais delas para ser visualizada por todos. A Figura 2 ilustra as interfaces do CP para o modo de instrutor à esquerda, utilizado pelo mestre, e para o modo de estudante à direita, utilizado pelos outros participantes. Algumas funcionalidades presentes no modo instrutor estão ausentes no modo estudante, como a visualização dos quadros da apresentação.

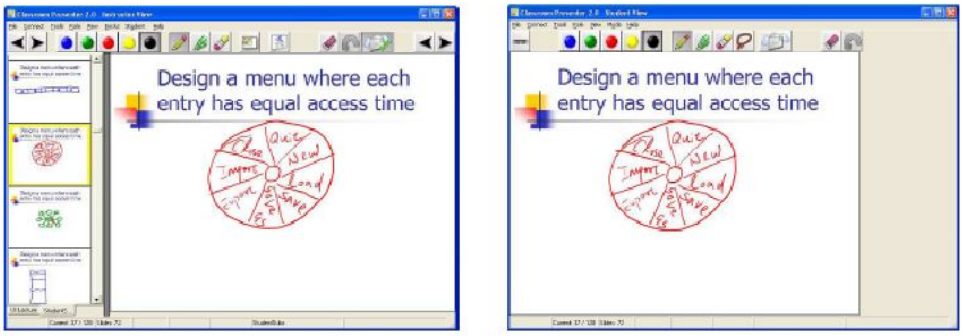

Figura 2 - O Classroom Presenter [Anderson, 2006]

Uma vantagem do CP é que este tem suporte para apresentações do tipo PowerPoint®, ferramenta de apresentação comumente utilizada em salas de aula. O desenvolvimento e aperfeiçoamento do CP fazem parte de um projeto que envolve também outras universidades norte-americanas, como a University of Virginia e a University of San Diego. Portanto, o CP já foi testado exaustivamente em várias disciplinas de cursos de graduação destas universidades, inclusive com excelentes índices de aceitação pelos usuários [Anderson, 2007]. Outra vantagem é a série de funcionalidades que o CP possui que melhoram a interatividade com o usuário, como:

- A opção de minimizar o quadro, aumentando a área de colaboração;
- O envio de perguntas de múltipla-escolha para os participantes. Estas perguntas são visualizadas pelos alunos que escolhem uma resposta e enviam de volta ao professor;

- A possibilidade de selecionar quais desenhos de um quadro um participante deseja enviar ao professor;
- É possível gravar uma sessão, e posteriormente visualizar passo-a-passo todas as anotações (contribuições) feitas, usando para isso a funcionalidade de Presenter Playback, ilustrado na Figura 3.

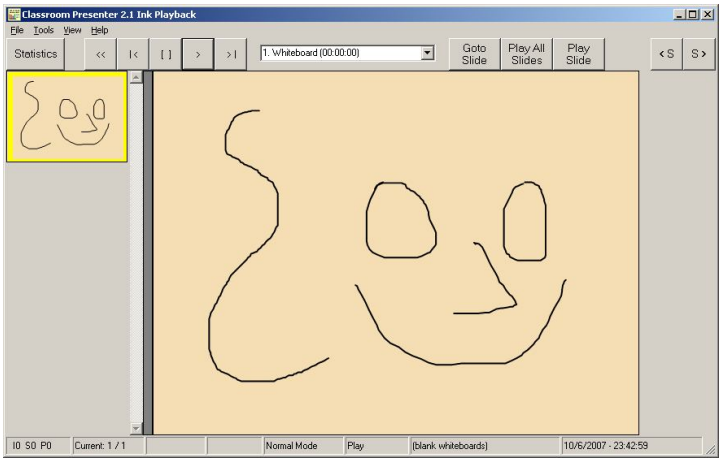

Figura 3 – O Presenter Playback

A maior limitação do CP é que esta ferramenta está restrita a *notebooks*, *tablet pcs* e computadores *desktops*, mais especificamente, apenas os dispositivos capazes de executar o .NET Framework. Ou seja, não existe uma versão para o uso em *handhelds*.

As pesquisas para evolução do CP continuam, e um dos trabalhos neste sentido é chamado de Ubiquitous Presenter (UP) [UbiquitousPresenter, 2007], que fornece uma interface *web* para os participantes, facilitando assim sua implantação. Segundo [Wilkerson, 2005], as grandes vantagens do UP são:

- Os participantes não precisam de um *tablet pc* para participar da aula. Tudo o que é necessário é um navegador *web* (*browser*) e a URL exportada pelo UP. A Figura 4 ilustra a visualização de um participante utilizando um *browser*.

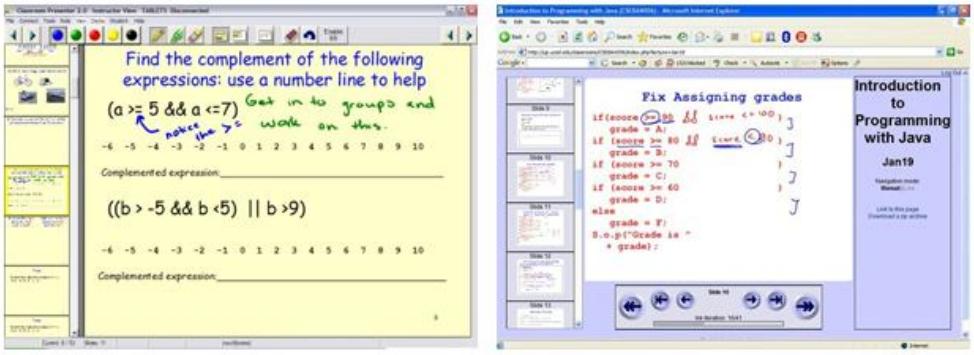

Figura 4 – Utilização do Ubiquitous Presenter [Wilkerson, 2005]

- O CP utiliza *multicast* para a colaboração entre os participantes, enquanto que o UP utiliza um *web service* para a comunicação. Esta arquitetura cliente-servidor facilita a implantação e utilização do UP se comparada com o CP, visto que este necessita de instalação e depende de *multicast*. A Figura 5 ilustra arquitetura do UP, onde os alunos só necessitam de um *browser* para participar da sessão de colaboração.

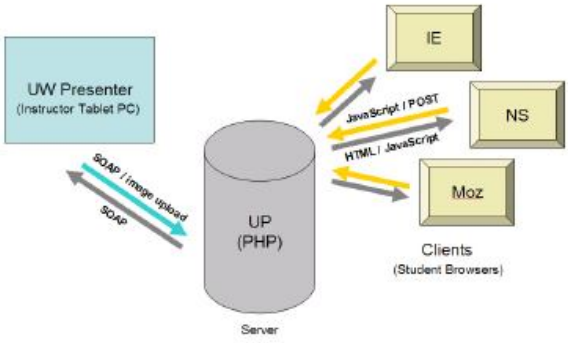

Figura 5 - A arquitetura do Ubiquitous Presenter [Wilkerson, 2005]

2.2. Livenotes

O Livenotes [Livenotes, 2007] é um sistema de compartilhamento de quadros eletrônicos com o objetivo de oferecer suporte à interação em pequenos grupos de estudo, não havendo diferença nos papéis exercidos pelos usuários. No Livenotes, qualquer contribuição de um participante é visualizada por todos os outros participantes do grupo. Em relação à conectividade do sistema, um dos

participantes exerce o papel de servidor, recebe as contribuições de outro participante e retransmite aos demais [Iles, 2002].

Este sistema possui como formas de contribuição o uso da caneta para desenhos e a inserção de textos, suporta a importação de apresentações PowerPoint®, adição de novos quadros eletrônicos, e outras funcionalidades como *zoom* e exportação da apresentação editada para documentos HTML [Kam, 2005]. Como visto na Figura 6, sua interface é bastante semelhante ao Classroom Presenter (CP). O Livenotes também foi desenvolvido para o uso em *tablet pcs*, e não possui versão para *handhelds*.

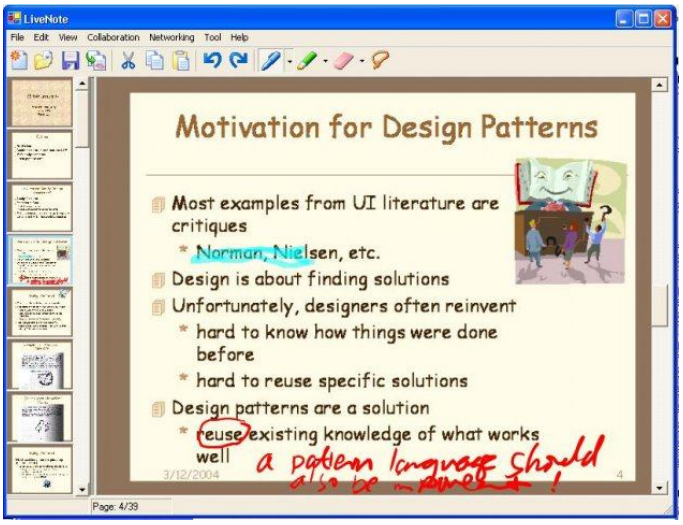

Figura 6 - O sistema Livenotes [Livenotes, 2007]

2.3. DyKnow

O DyKnow (*Dynamic Knowledge Transfer*) [DyKnow, 2007] é um sistema comercial para apresentações colaborativas, que oferece suporte a apresentações PowerPoint®, adição e remoção de quadros durante a apresentação, co-edição dos quadros pelos participantes, e diversas outras funcionalidades que o diferencia dos demais sistemas, como [Berque, 2004]:

- Além do compartilhamento de desenhos, os participantes podem compartilhar textos, imagens e páginas *web*. Existem também opções de *zoom*, desfazer e refazer, copiar e colar, alinhamento de texto, entre outros;

- Todos os participantes possuem uma área para anotações particulares;
- O participante controlador pode enviar enquetes (questionários eletrônicos) e questões de múltiplas escolhas aos participantes, e obter gráficos e tabelas a partir de suas respostas;
- O controlador pode visualizar o *status* de cada um dos participantes da aula: verificar se estão acompanhando o quadro atual, verificar se estão visualizando outro quadro, ou se algum participante desconectou. Além disso, o controlador pode visualizar o quadro atual de qualquer participante conectado, para saber o que este está desenhando;
- O controlador pode bloquear aplicações utilizadas pelos outros participantes como aplicações de troca de mensagens instantâneas e jogos, e também pode bloquear o *mouse* e o teclado, evitando qualquer ação do participante. Em qualquer momento, o controlador pode visualizar um *screenshot* da tela de qualquer participante, para saber quais participantes estão distraídos e enviar mensagens de alerta a eles;
- O DyKnow possui seu próprio sistema de troca de mensagens, utilizado para esclarecer dúvidas entre os participantes e o controlador, de modo que não atrapalhe o desenvolvimento da aula;
- As apresentações podem ser registradas e os participantes podem rever passo-a-passo todas as interações ocorridas durante uma apresentação compartilhada. Neste modo de *replay*, desenho por desenho, é mais fácil para um estudante entender como a apresentação evoluiu.

A Figura 7 ilustra a interface do DyKnow com as opções de colaboração como caneta e texto, o quadro compartilhado que está sendo editado, e a área de anotações particulares, entre outras.

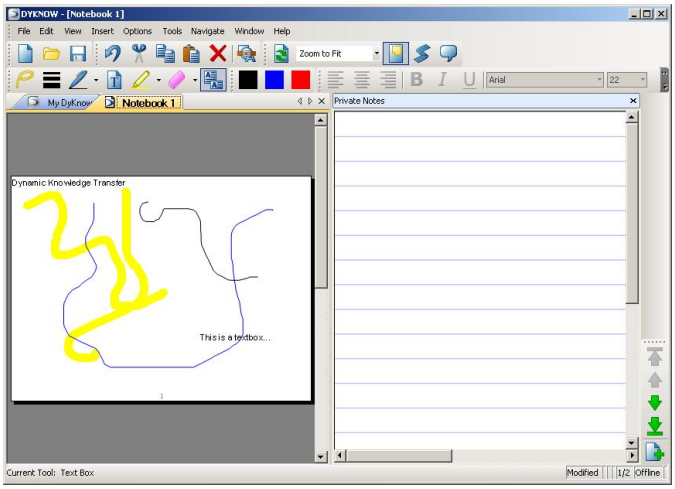

Figura 7 - O sistema DyKnow (Dynamic Knowledge Transfer)

O DyKnow não possui versão para o uso em *handhelds*, sendo específico para computadores *desktops*, *notebooks* e *tablet pcs*, e é a evolução de um projeto acadêmico conhecido como DEBBIE (*DePauw Electronic Black Board for Interactive Education*).

2.4. Teamspace/MeetingClient

O Teamspace [Geyer, 2001] é um sistema para colaboração que objetiva o gerenciamento de processos compartilhados e manutenção de artefatos distribuídos. Este sistema oferece um maior controle sobre os processos de um projeto que envolva vários grupos distribuídos em locais remotos. Durante uma reunião de grupos para discussão de tópicos a respeito de determinados assuntos, é utilizada a aplicação MeetingClient, que se integra com o Teamspace e suas outras aplicações, como agenda e lista de tarefas do projeto.

O MeetingClient [Richter, 2001] oferece basicamente o mesmo que os outros sistemas aqui relatados oferecem: compartilhamento de apresentações PowerPoint® ou quadros eletrônicos e a possibilidade de contribuição a partir de caneta ou texto. Na Figura 8 é ilustrado a interface do MeetingClient.

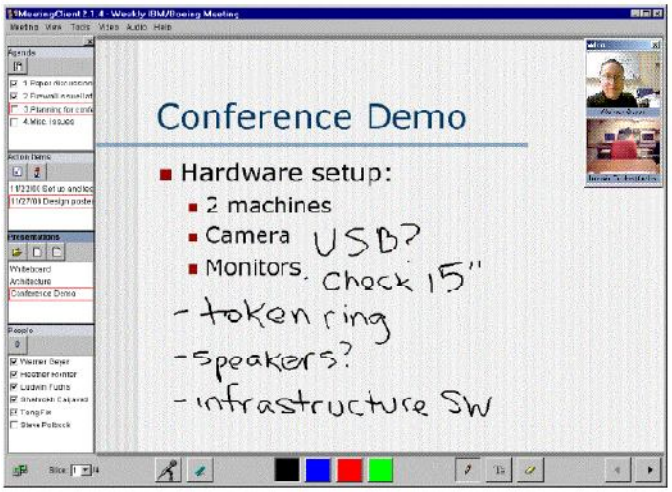

Figura 8 - O MeetingClient/Teamspace [Geyer, 2001]

Como parte do Teamspace, cada sessão para discussão de um assunto através do MeetingClient se torna um artefato do projeto discutido. Assim, qualquer participante poderá rever o que foi discutido através do aplicativo MeetingViewer. Este aplicativo exibe passo-a-passo todas as colaborações realizadas durante a sessão colaborativa. A Figura 9 ilustra o MeetingViewer.

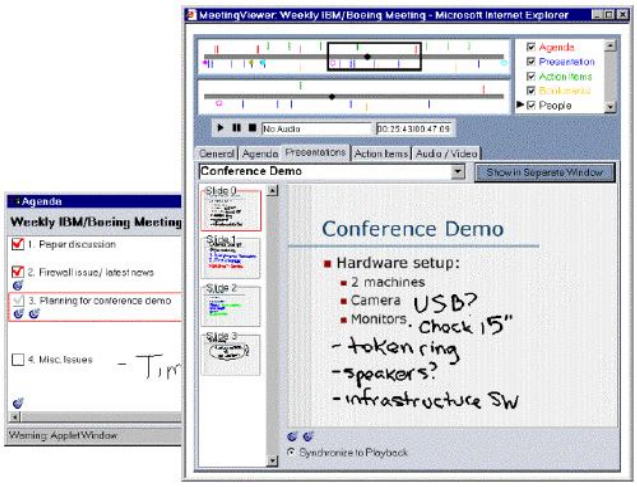

Figura 9 – Visualização do MeetingViewer [Geyer, 2001]

2.5. Virtual Multiboard

O Virtual Multiboard (VMB) é um sistema que provê gerenciamento das aplicações utilizadas em uma apresentação, e funcionalidades de anotação e

gravação de uma aula [Rößling, 2004]. Ao contrário das outras aplicações que fazem uso diretamente das apresentações PowerPoint®, exportando os quadros destas a um formato próprio para utilização, o VMB executa paralelamente uma apresentação PowerPoint®, ou qualquer outro tipo de apresentação. Devido este desacoplamento das apresentações utilizadas, as anotações feitas durante a apresentação são inseridas em um painel transparente (*glass pane*) do próprio VMB. A partir de seus *notebooks* ou *palmtops*, os participantes podem enviar perguntas ao controlador e participar de questionários eletrônicos.

O VMB oferece suporte à visualização de múltiplos quadros de uma apresentação simultaneamente, pois considera que os participantes de uma apresentação podem perder a linha de aprendizado caso visualizem sempre um único quadro. Foram desenvolvidas aulas e palestras onde os últimos dois quadros da apresentação eram exibidos juntamente com o quadro atual, ajudando os participantes a entenderem melhor o conteúdo da apresentação.

2.6. Tablet Mylar Slides

Apesar de não ser uma aplicação colaborativa, o Tablet Mylar Slides (TMS) [TabletMylarSlides, 2007] é muito semelhante à aplicação proposta, pois trabalha com quadros eletrônicos, o suporte a apresentações PowerPoint®, e a possibilidade de realizar contribuições com desenhos. O uso do TMS é semelhante à maneira que o PowerPoint® é utilizado nas salas de aula: o professor utiliza um computador conectado a um projetor, que exibe a apresentação para os participantes da aula.

De acordo com [Golub, 2004], o diferencial do TMS para o PowerPoint® é a altura dos quadros, que pode ser infinita, facilitando a contribuição do professor, visto que este sempre terá espaço para fazer suas anotações; a exportação da apresentação editada para formato HTML, de modo que possa ser distribuída aos participantes que estiveram ausentes; e o conceito de caneta secreta (*hidden ink)*. Com esta caneta o professor realiza suas anotações e estas não são exibidas pelo projetor. A Figura 10 ilustra um exemplo do uso do TMS. À esquerda, o TMS é utilizado pelo professor para provar por indução uma fórmula matemática, e à direita é mostrado o que é exibido pelo projetor. Pode-se notar que parte do que

foi escrito pelo professor não é visualizado pelos participantes da aula, pois o mesmo utilizou a caneta secreta para fazer suas anotações particulares.

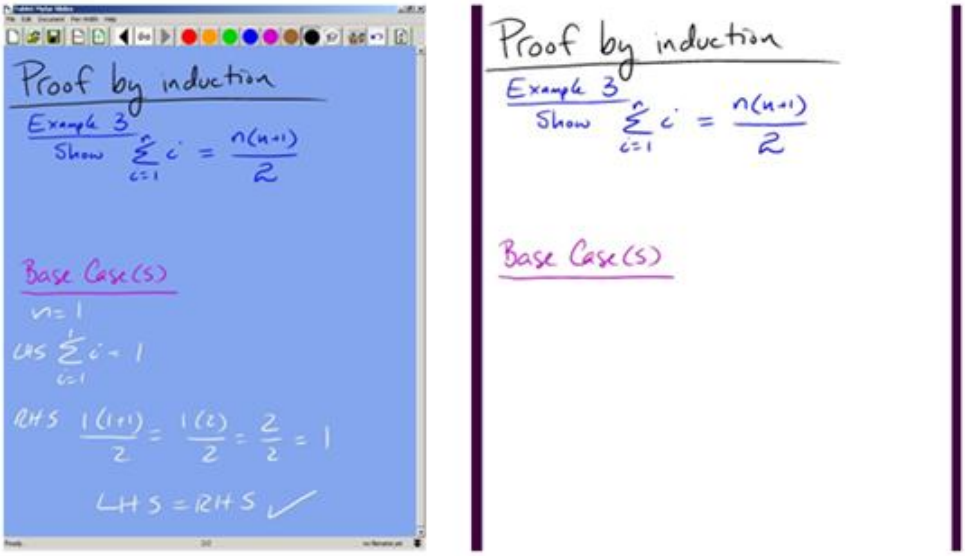

Figura 10 – Caneta secreta do TMS [TabletMylarSlides, 2007]

2.7. IdeaLink

O IdeaLink [Salber, 2001] é um sistema de colaboração avançada desenvolvido para utilização em *pamltops*. Este sistema oferece aos usuários espaços virtuais colaborativos para comunicação entre participantes de uma sessão. Um participante pode colaborar com ferramentas de desenho ou texto, escolhendo suas cores e tamanhos, além de poder enviar mensagens instantâneas para outros participantes da sessão. Através de múltiplos canais, um usuário pode participar de várias sessões de colaboração simultaneamente. Não existe suporte a apresentações PowerPoint® ou outro tipo de apresentação. O quadro eletrônico onde a colaboração acontece tem tamanho ajustável, porém ele é único para cada sessão.

A arquitetura sobre o IdeaLink é chamada de Handy Andy, e oferece aos usuários a capacidade de trabalhar independente de qual dispositivo está sendo utilizando. Esta arquitetura foi desenvolvida para superar os problemas da computação ubíqua e de infra-estrutura de redes sem-fio, transferindo a aplicação para um servidor de rede. Problemas provenientes de redes sem-fio, como

conexões perdidas e largura de banda limitada, são automaticamente tratadas pela arquitetura. A arquitetura cliente-servidor possibilita a um participante trocar o dispositivo que esteja utilizando e continuar participando da sessão de colaboração [Smailagic, 2001].

2.8. SharedPad

O SharedPad é uma aplicação protótipo que disponibiliza a usuários um quadro eletrônico para edição [Kadous, 2004]. Este sistema é extremamente simples e foi desenvolvido durante os testes da MICA (*Multimodal Interagent Communication Architecture*), um *middleware* para computação ubíqua que trabalha com o conceito de memória global compartilhada, que funciona tanto como um mecanismo de armazenamento quanto de comunicação. O SharedPad pode ser executado em computadores *desktops* e *palmtops*, e é bastante limitado se comparado com os outros trabalhos apresentados.

2.9. Quadro Comparativo

Cada um dos trabalhos relacionados pretende oferecer aos usuários uma experiência de aprendizado ativa, aumentando a produtividade dos participantes e incentivando o debate entre eles. Para efeito de comparação dos trabalhos, as seguintes características foram utilizadas:

- O modo de edição e colaboração das apresentações: quais são as ferramentas utilizadas para prover interação entre os participantes como caneta, texto, envio de enquetes, entre outros;
- O modelo de colaboração: a aplicação suporta o paradigma de uma sala de aula, onde um participante faz o papel do professor controlando o andamento da apresentação, ou o modelo de debate de pequenos grupos de participantes;
- O suporte a apresentações;
- A possibilidade de fazer anotações particulares;
- A execução em *handhelds.*

A Tabela 1 apresenta essas informações para cada um dos trabalhos relacionados.

Tabela 1 - Quadro comparativo entre os trabalhos relacionados

Aplicativo	**Modo de edição e colaboração**	**Modelo de colaboração**	**Suporta apresentações**	**Anotações particulares**	**Suporta *handhelds***
Classroom Presenter	Caneta, enquetes, envio de colaborações para o professor	Sala de aula	Sim, adição de quadros e apresentações PowerPoint®	Sim	Não
Livenotes	Caneta e texto	Debate entre participantes de um grupo	Sim, adição de quadros e apresentações PowerPoint®	Não	Não
DyKnow	Caneta, texto, envio e requisição das colaborações entre participantes, compartilhamento de figuras, páginas web	Sala de aula	Sim, adição de quadros e apresentações PowerPoint®	Sim	Não
Meeting Client	Caneta e texto	Debate entre participantes de um grupo	Sim, adição de quadros e apresentações PowerPoint®	Não	Não
Virtual Multiboard	Caneta e formas geométricas, como retângulos, envio de perguntas	Sala de aula	Sim, apresentações PowerPoint® ou outros	Não	Sim
Tablet Mylar Slides (TMS)	Caneta com a opção da caneta secreta (*hidden ink*)	Não se aplica	Sim, adição de quadros e apresentações PowerPoint®	Sim	Não

IdeaLink	Caneta e texto	Debate entre participantes de um grupo	Não, quadros eletrônicos fixos para cada canal	Não	Sim
SharedPad	Caneta	Debate entre participantes de um grupo	Não	Não	Sim

3
Fundamentação Conceitual

Uma **apresentação colaborativa** é uma apresentação co-editada de forma organizada por vários usuários de posse de dispositivos computacionais móveis no contexto de uma aula, uma palestra ou de uma reunião, permitindo um nível maior de interatividade entre os participantes do que uma apresentação comum [Maite, 2003]. O uso bem aplicado da tecnologia é imprescindível para o sucesso de uma apresentação colaborativa, onde eventos indesejados como falhas na comunicação (p. ex. desconexão de um participante) ou problemas na usabilidade do aplicativo prejudicam de modo sensível o andamento da apresentação [Beavers, 2004].

O processo de uma apresentação colaborativa está vinculado ao conceito de uma **sessão de colaboração**, que consiste de um grupo de **participantes** (i.e. representados pelos seus dispositivos móveis com interface de rede sem fio), dos quais: um deles executa o papel de controlador e seu dispositivo será doravante denominado **mestre**; um ou mais participantes executam o papel de **contribuidor**, e acompanham e colaboram com a apresentação; e um ou mais dispositivos, provavelmente conectados a projetores, fazem o papel de **visualizadores**. O ponto focal de uma sessão de colaboração é uma **apresentação compartilhada**, que é constituída por **quadros**. O mestre é o único dispositivo com o **controle de visualização da apresentação** pelos demais participantes, isto é, o controle sobre qual quadro será mostrado nos dispositivos dos demais participantes, bem como da aceitação (ou rejeição) das contribuições dos participantes. Cada contribuidor poderá adicionar desenhos e/ou textos a um quadro, mover o cursor, apagar e corrigir os seus desenhos/textos, bem como submeter modificações locais (feitas no quadro corrente) para o participante mestre que, se desejar, poderá exibir a **contribuição do participante** para os demais participantes, através dos visualizadores. Os visualizadores possuem todos os quadros que o mestre possui, exceto seus quadros particulares, quadros adicionados que servem para anotações pessoais do mestre. O conjunto de todas as contribuições dos contribuidores só poderá ser visualizado no dispositivo mestre. Os demais contribuidores só

poderão visualizar as suas edições locais no quadro corrente. O quadro corrente é definido pelo mestre, e é mostrado em todos os dispositivos dos participantes, independentemente de suas modificações locais. A qualquer momento um participante pode se juntar ao grupo, ou sair dele. Quando se junta ao grupo, um participante pode receber a versão atual da apresentação compartilhada. A Figura 11 ilustra uma sessão de colaboração onde o mestre envia aos outros participantes os quadros da apresentação que irá ser compartilhada, comandos de sincronização destes quadros, e suas edições nos mesmos.

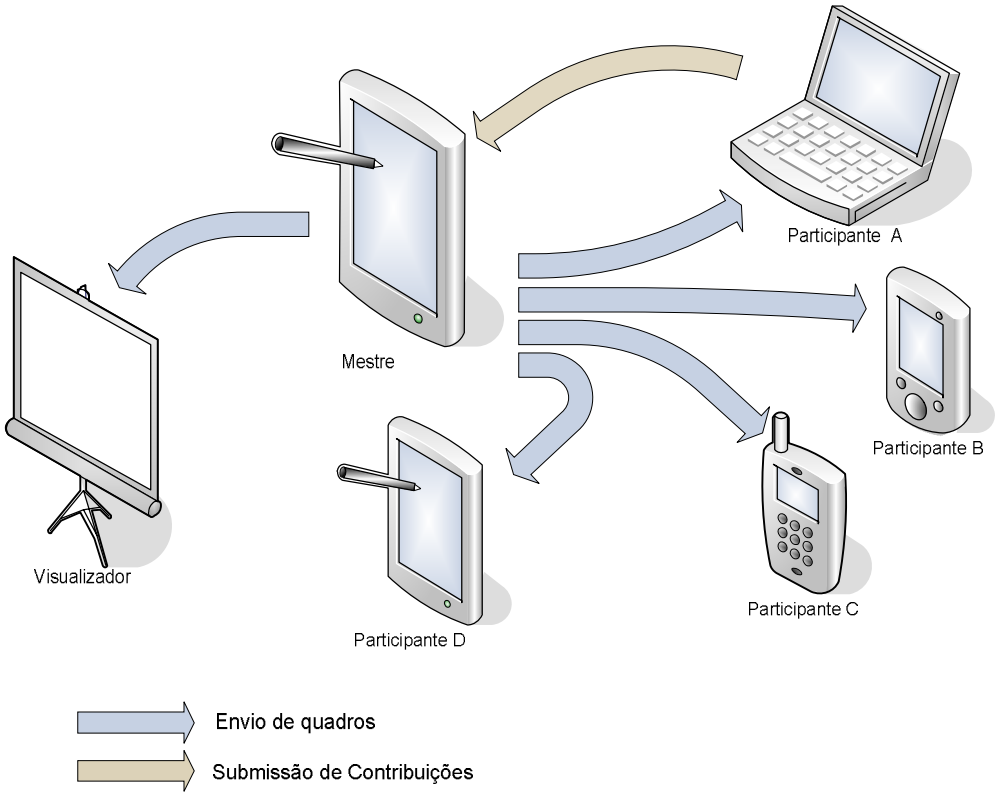

Figura 11 - Sessão de colaboração entre dispositivos heterogêneos

O mestre pode controlar o que é exibido nos visualizadores, decidindo se mantém uma sincronização com estes ou não. Desta maneira, o mestre pode fixar um quadro para exibição, e posteriormente requisitar uma nova sincronização com os mesmos. Quando um contribuidor realiza uma nova contribuição, este perde a sincronização com o mestre automaticamente. Isto acontece para evitar assim a mudança de seu quadro atual durante a realização de uma contribuição. Em

qualquer momento este contribuidor pode recuperar a sincronização com o mestre, assim como o mestre também pode requisitar a sincronização de todos os contribuidores.

O conjunto de quadros compartilhados pelo mestre é chamado de ***deck*.** O mestre trabalhará sempre com dois *decks*: um que contém os quadros da apresentação, com a possibilidade de adição e remoção de quadros; e o segundo com as contribuições enviadas pelos contribuidores.

O modelo de colaboração utilizado para o desenvolvimento da solução proposta é baseado no modelo de colaboração do Classroom Presenter [Anderson, 2004a]. Ligeiras modificações foram realizadas para atender os objetivos da aplicação.

3.1. Contexto e Aplicações sensíveis a contexto

A aplicação proposta neste trabalho, além de oferecer suporte a apresentações colaborativas, é sensível a informações de contexto computacional, mesmo que de forma experimental. É importante definir o que é uma informação de contexto e como utilizar essas informações para definir o comportamento da ferramenta. Segundo [Dey, 2000], contexto é qualquer informação que possa ser usada para caracterizar a situação de uma entidade, onde entidade é uma pessoa, lugar, ou objeto relevante entre o usuário e a aplicação, incluindo o usuário e aplicação em si. Na aplicação proposta, o foco será apenas em contexto computacional, ou seja, informações relacionadas às características físicas e lógicas dos dispositivos como localidade, capacidade de processamento, memória, entre outras.

Informações de contexto podem ser coletadas de maneira implícita, como a descoberta da localização do usuário a partir de algum mecanismo e propriedades físicas do dispositivo como memória livre ou energia restante; ou explícita, quando o usuário é questionado de algo como sua identidade.

O principal objetivo da computação sensível a contexto é melhorar a interação homem-máquina realizando adaptações necessárias a partir de informações relevantes para o usuário e para a aplicação. É importante definir qual a relevância das informações: o que é importante para uma aplicação pode

não ter valor para outra. Por exemplo, a localização é importante quando uma aplicação utiliza medidas como altura ou peso, que variam de país para país. A identidade do usuário não tem significado para a mesma aplicação, o que não é o caso de um *site* de comércio eletrônico que necessita desta identidade para poder efetuar vendas e fazer ofertas personalizadas de acordo com o histórico de compras do usuário.

Segundo [Coutaz, 2005], uma aplicação é sensível a contexto se utiliza essas informações para prover ao usuário informações ou serviços relevantes de acordo com a tarefa que está sendo exercida pelo mesmo. Portanto, aplicações que apenas acessam informações de contexto também podem ser consideradas sensíveis a contexto.

Acredita-se que uma aplicação para apresentações colaborativas sensível a informações de contexto computacional dispõe de alternativas para prover uma melhor interatividade entre os participantes, e um melhor controle das apresentações pelo mestre. Por exemplo, os seguintes cenários motivam a utilização deste tipo de contexto:

- Durante uma apresentação, o mestre pode definir que somente contribuidores presentes em uma determinada **área/localidade** poderão submeter contribuições a ele. Caso um contribuidor se locomova para fora da área determinada, este perde a opção de enviar suas contribuições;
- A percepção da perda e retomada da **conexão** pelos contribuidores pode disparar automaticamente uma função de envio dos quadros que os participantes não receberam quando estavam desconectados;
- A informação sobre pouca **energia do dispositivo** participante pode gerar um alerta no mestre (ou no próprio participante), para que este envie logo a sua contribuição e conecte-se na tomada;
- A detecção de uma **fraca conectividade** de um ou mais participantes pode ser utilizado para decisão sobre enviar de uma vez todo o *deck* ou enviar quadro a quadro, e assim reduzir as chances de perdas de mensagens na rede, ou então usar uma função para reduzir qualidade das imagens dos quadros.

4
Questões de Middleware

Durante o planejamento da aplicação proposta, questões relacionadas à sua propriedade colaborativa, à edição de quadros de uma apresentação e ao acesso a informações de contexto, tiveram que ser resolvidas para facilitar o desenvolvimento da mesma. O principal objetivo foi escolher ou desenvolver componentes independentes que pudessem ser re-utilizados em quaisquer outros projetos.

4.1.
Colaboração

A colaboração entre os usuários da aplicação proposta envolve detalhes de nível de rede como: a maneira como a conexão será realizada; o envio e recebimento de mensagens; e o tratamento de possíveis falhas durante a comunicação. Esses detalhes requerem um grande esforço por parte dos desenvolvedores de uma aplicação colaborativa, já que ao invés de estarem concentrados no desenvolvimento da aplicação em si, os desenvolvedores perdem um tempo razoável no desenvolvimento desta comunicação.

A escolha de um *middleware* que encapsule as funcionalidades da comunicação é uma forma natural de facilitar o desenvolvimento de uma aplicação colaborativa. Outras vantagens são a confiabilidade do *middleware*, que pode ter sido utilizado em outros projetos com sucesso e a agilidade no processo de desenvolvimento da aplicação, visto que todas as funcionalidades necessárias à comunicação estarão contidas no *middleware*.

4.1.1.
Motivação da Escolha do ConferenceXP

O ConferenceXP [ConferenceXP, 2006] foi escolhido como *middleware* de colaboração da aplicação proposta devido a facilidade de desenvolver aplicações colaborativas com o mesmo. Desenvolvedores podem criar aplicações

colaborativas ou adaptar aplicações já existentes utilizando as *APIs* e o conjunto de classes básicas do ConferenceXP. Este *middleware* facilita a tarefa dos desenvolvedores, pois trata questões relacionadas à comunicação de aplicaçãoes que devem colaborar entre si. A Figura 12 exibe a arquitetura de uma aplicação colaborativa que utiliza o ConferenceXP, dividindo a mesma em dois níveis: nível de rede, onde o ConferenceXP realiza a comunicação; e o nível de aplicação, onde são realizadas as funcionalidades principais da aplicação.

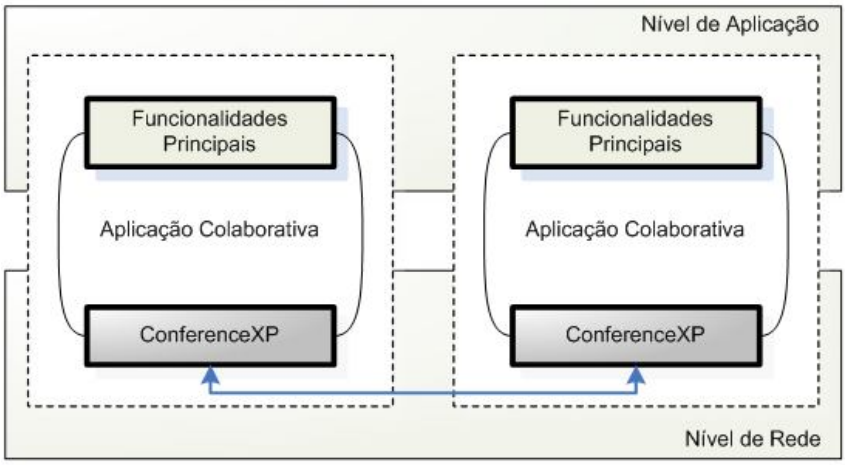

Figura 12 – Aplicação colaborativa utilizando o ConferenceXP

O ConferenceXP foi desenvolvido para a plataforma .NET Framework [NET, 2007], na linguagem C# [C#, 2007], mesma linguagem de desenvolvimento da aplicação proposta, o que facilita a integração e o aprendizado do mesmo. Este *middleware* emprega uma arquitetura *peer-to-peer* ao utilizar *multicast*, não havendo necessidade de um servidor que receba e repasse todas as mensagens, o que facilita a implantação de qualquer aplicação. O uso de *multicast* ao invés de *unicast* garante um menor consumo de recursos de rede, como visto na Figura 13 [Multicast, 2007].

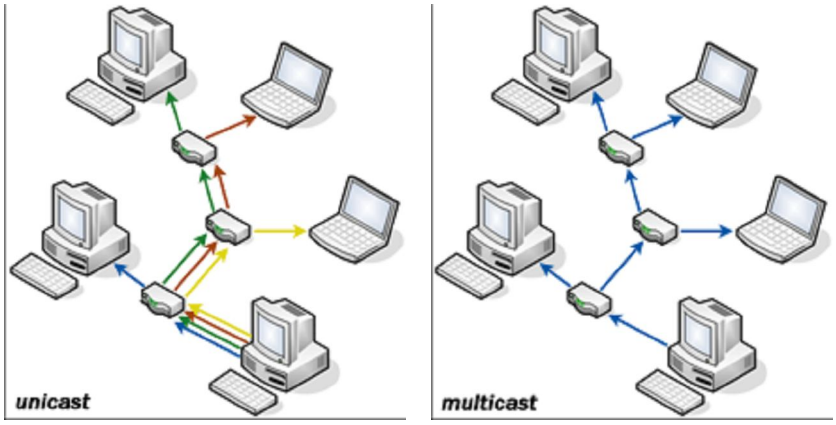

Figura 13 - Diferenças ente o *unicast* e o *multicast* [Multicast, 2007]

O Classroom Presenter (vide seção Classroom Presenter2.1) foi desenvolvido utilizando o *middleware* ConferenceXP, o que comprova sua eficiência e confiabilidade no desenvolvimento de aplicações colaborativas.

4.1.2. Arquitetura do ConferenceXP

O ConferenceXP é dividido em quatro camadas lógicas: Network Transport, ConferenceAPI, ConfereceXP Capability e ConferenceXP Application. A Figura 14 ilustra esta arquitetura do ConferenceXP.

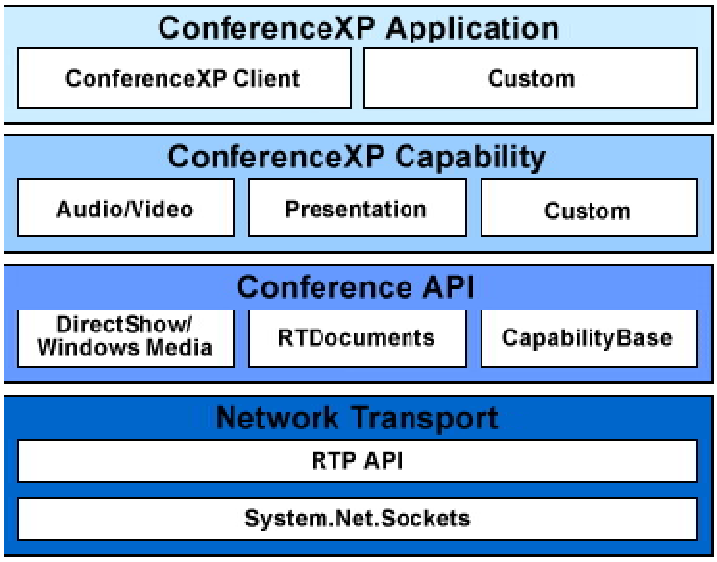

Figura 14 - A arquitetura do ConferenceXP [ConferenceXP, 2006]

A camada **Network Transport** oferece uma implementação do protocolo RTP (Real-Time Transport Protocol) [RTP, 2007], baseado na implementação dos Windows Sockets. Este protocolo é um padrão da IETF (Internet Engineering Task Force) [IETF, 2007] para transmissão de áudio e vídeo em redes *peer-to-peer*, e foi desenvolvido para ambientes onde é necessário um baixo nível de latência. Para prevenir perda de dados, o ConferenceXP implementa algoritmos FEC (Forward Error Correction).

Na camada **ConferenceAPI**, os desenvolvedores podem criar aplicações colaborativas ou *capabilities*, sem se preocupar com aspectos de rede. A classe *CapabilityBase* encapsula a funcionalidade requerida de outras partes da camada de conferência e serve como base para criação de novas *capabilities*, que são componentes que oferecem funcionalidades às aplicações do ConferenceXP. Com a API RTDocument, aplicações e *capabilities* utilizam um protocolo para transferência de documentos e *inks*.

Na camada **ConferenceXP Capability**, originalmente existem duas *capabilities*: a de apresentação (*Presentation*), que oferece suporte à apresentações PowerPoint® e ao uso de *ink*, e a de áudio e vídeo, utilizada para conferências. Nesta camada estão as *capabilities* customizadas desenvolvidas para aplicações colaborativas, que junto com a camada **ConferenceXP Application,** oferecem interfaces aos usuários.

4.1.3. Limitações e o .NET Compact Framework

Para entender melhor as limitações do ConferenceXP, é necessário primeiro distinguir as diferenças entre o .NET Framework e o .NET Compact Framework [CFNET, 2007a]. O .NET Compact Framework (.NET CF) é um subconjunto do .NET Framework que oferece interoperabilidade com o sistema operacional Windows Mobile de um *handheld* como um *palmtop* ou um *smartphone* [CFNET, 2007b]. Ao comparar as duas plataformas de desenvolvimento, a arquitetura do .NET CF é bastante limitada, e não possui diversas funcionalidades, como o suporte nativo a serialização de objetos. Este recurso é essencial para aplicativos de colaboração, onde objetos são enviados e recebidos através da rede [CFNET, 2007c].

Outra limitação é o tratamento dado à coleta de *ink*. O ConferenceXP utiliza um componente desenvolvido para *tablet pcs*, chamado **Microsof.Ink**. Este componente é utilizado no desenvolvimento de aplicações que utilizem a caneta do *tablet pc* para captura de desenhos e reconhecimento de textos. O Microsoft.Ink é restrito ao .NET Framework, e, portanto, não pode ser utilizado em uma aplicação que deva executar em dispositivos mais limitados como *palmtops*.

Estas limitações do .NET CF em relação ao .NET Framework refletem as limitações do ConferenceXP, que não foi desenvolvido para ser utilizado em dispositivos que utilizam o .NET CF.

4.1.4. Adaptações Necessárias

O ConferenceXP faz uso de vários métodos e classes que só existem no .NET Framework, não possuindo iguais correspondentes no .NET CF. Logo, uma série de adaptações foi necessária para que fosse possível sua utilização em ambas as plataformas, atendendo assim os requisitos da aplicação proposta.

É importante notar que todas as aplicações desenvolvidas para o .NET CF que utilizem somente as funcionalidades do próprio *framework*, ou seja, não utilizem funcionalidades nativas dos dispositivos portáteis utilizados, podem ser executadas no .NET Framework. Assim, o trabalho de adaptação do ConferenceXP foi realizado da seguinte maneira: a partir do código-fonte das *APIs*, foram realizadas alterações para compilação e execução no .NET CF. Algumas classes tiveram que ser implementadas para preservar as funcionalidades originais do *middleware*, como: *DicitonaryBase*, classe abstrata que serve como base para coleções de pares chave/valor; e *SynchronizedQueue*, implementação de uma fila sincronizada. Várias alterações em métodos utilizados no código destas *APIs* também foram necessárias durante o trabalho de adaptação.

Também foram utilizadas as bibliotecas do OpenNETCF [OpenNETCF, 2007], para funcionalidades como o acesso a arquivos e configuração e métodos de reflexão. O OpenNETCF é um projeto apoiado pela Microsoft®, e tem como objetivo desenvolver funcionalidades que não existem no .NET CF.

4.1.4.1. Serialização de Objetos

Como o .NET CF não possui suporte nativo a serialização de objetos, foi utilizado um projeto independente chamado CompactFormatter [CompactFormatter, 2007], que é um formatador genérico para o .NET CF capaz de serializar a maioria dos objetos utilizando reflexão, permitindo redefinir os algoritmos de serialização se necessário. A interface disponibilizada por este componente é bastante similar à do *BinaryFormatter* e do *SOAPFormatter*, formatadores utilizados comumente no .NET Framework. É válido mencionar que a serialização entre as diferentes plataformas também é complexa devido ao fato de que um mesmo objeto pode possuir diferentes implementações em cada plataforma. Por exemplo, a classe *Hashtable* possui atributos diferentes na versão para .NET Framework em relação à versão para .NET CF. Para a maioria das classes que se encaixam neste perfil, o CompactFormatter utiliza algoritmos que provêm uma solução automática.

A Figura 15 exibe a comunicação entre aplicações em diferentes plataformas. Toda serialização de objetos será feita utilizando o CompactFormatter, mesmo quando ambas as aplicações estiverem sendo executadas no .NET Framework.

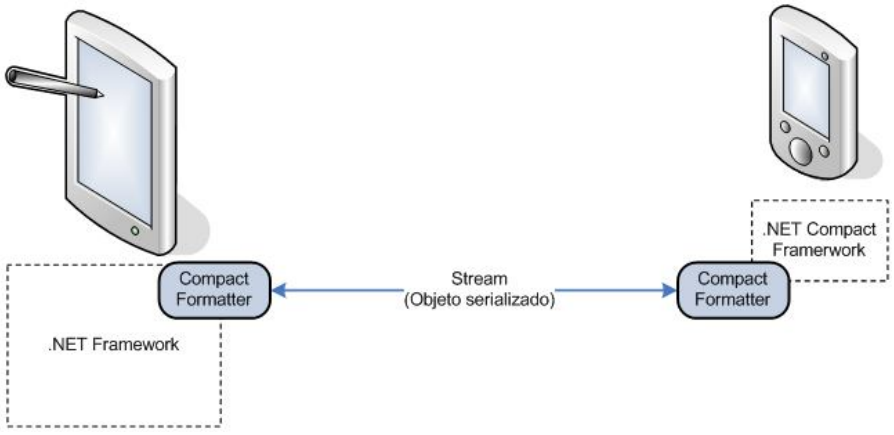

Figura 15 - A serialização de objetos utilizando o CompactFormatter

4.1.4.2. Componente de Edição

O ConferenceXP utiliza o componente Microsof.Ink para a captura de desenhos feitos com a caneta do *tablet pc*. Este mesmo componente é utilizado pelo Classroom Presenter para a edição de quadros, e, portanto não pode ser usado para o desenvolvimento da aplicação proposta. Logo, foi necessário desenvolver um componente que capturasse desenhos realizados em qualquer dispositivo, seja este um *notebook*, um *tablet pc* ou um *handheld*, e oferecesse a adição de textos.

O componente desenvolvido foi chamado de **LAC.Contribs**, e substituiu todas as referências ao componente Microsof.Ink no código adaptado do ConferenceXP. Apesar de possuir uma qualidade inferior em relação ao componente originalmente utilizado, o desenho da *ink* é satisfatório, mesmo quando visualizado em diferentes dispositivos. A Figura 16 exibe um exemplo de uma imagem desenhada em um *palmtop*, e sua visualização em um *desktop*.

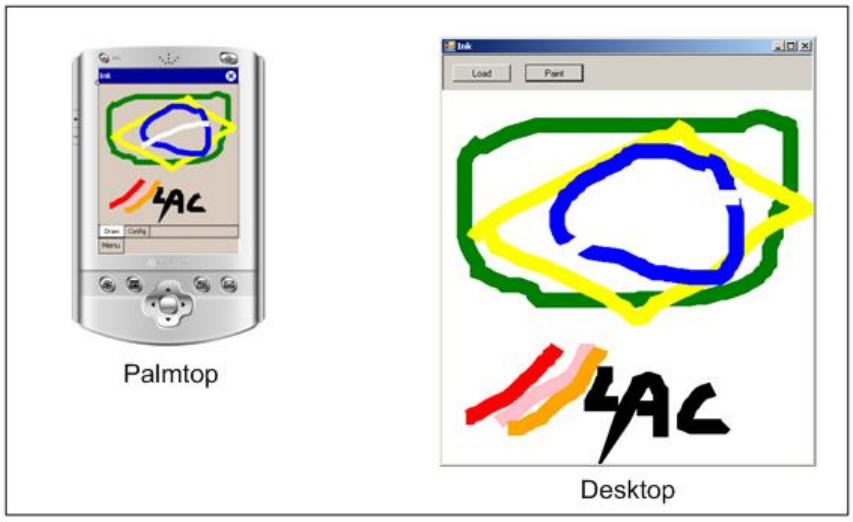

Figura 16 - Desenho feito em um *palmtop* e visualizado em um *desktop*

O LAC.Contribs foi desenvolvido de maneira independente, podendo ser utilizado por quaisquer aplicações que necessitem capturar desenhos ou adicionar textos em ambas as plataformas .NET. Para que uma aplicação possa fazer uso deste componente, somente é necessário que o controle visual definido como área de desenho herde da classe *ComponentBaseControl*. A Figura 17 exibe o diagrama de classes do componente LAC.Contribs.

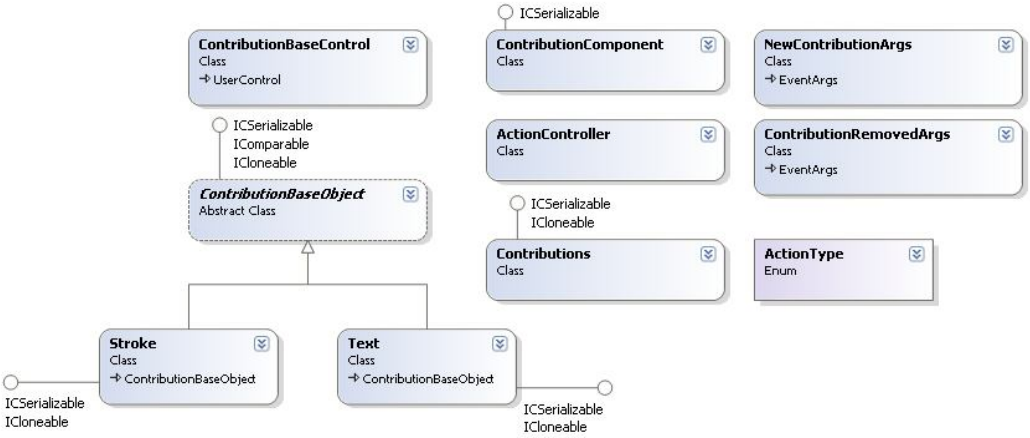

Figura 17 - O diagrama de classes do componente LAC.Contribs

Após a ativação do componente, a partir de um atributo do tipo ActionType, é possível escolher o modo de captura: *ink* ou texto. A *ink* é desenhada a partir das coordenadas de captura da caneta ou do *mouse*, e é possível formatar sua cor e largura. Para o texto, o usuário escolhe a fonte, a cor e o tamanho e clica no ponto onde deseja inserir o texto. Então, uma janela é visualizada e solicita ao usuário o que deve ser escrito. O texto será desenhado no controle visual, como visto na Figura 18.

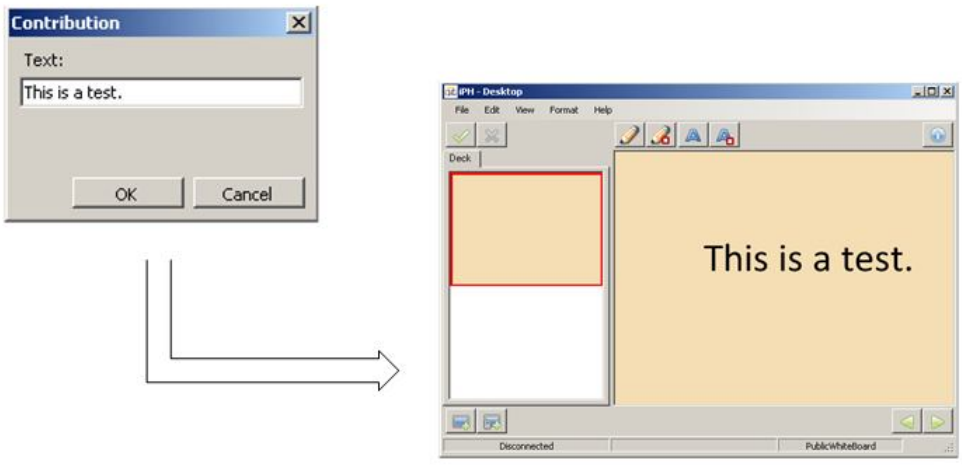

Figura 18 - Inserindo um texto com o componente LAC.Contribs

Com o LAC.Contribs, é possível também apagar os desenhos realizados e os textos inseridos. Ao escolher este modo de borracha, basta que o usuário clique em uma linha desenhada ou em um texto e os mesmos são apagados.

Uma linha desenhada é um objeto da classe *Stroke*, e um texto inserido é um objeto da classe *Text*. Ambas as classes derivam de uma classe abstrata chamada

ContributionBaseObject, e são serializáveis pelo componente CompactFormatter. A classe *ContributionBaseObject* possui atributos de identificação únicos e marcadores de tempo. Os atributos de identificação devem ser únicos, pois para uma aplicação como a proposta neste trabalho, é ser necessário saber, por exemplo, qual das contribuições foi apagada. Assim, a aplicação pode enviar aos demais participantes comandos para exclusão da determinada contribuição. Os marcadores de tempo são atribuídos no momento em que cada objeto é criado, e são necessários para a ordem do desenho dos objetos. Isto garante que não haverá sobreposição incorreta entre os mesmos.

O LAC.Contribs oferece também suporte a eventos disparados pelas ações de criação e remoção de objetos, sejam eles desenhos ou textos. Uma aplicação talvez necessite saber quando um desenho foi realizado, e qual o objeto resultante da ação. Por exemplo, a aplicação proposta necessita saber quando um evento deste tipo ocorreu, pois é necessário enviar aos demais participantes as novas contribuições realizadas pelo mestre.

4.1.5. CompactConferenceXP

Ao adaptar as *APIs* do ConferenceXP para execução também no .NET CF, foram utilizados diversos componentes como o OpenNETCF, o CompactFormatter e o LAC.Contribs, além de várias alterações necessárias. O resultado destas adaptações é o conjunto de *APIs* chamado de **CompactConferenceXP**. Qualquer aplicação que necessite de colaboração entre as diferentes plataformas pode utilizar estas *APIs* para este fim. É importante ressaltar que outras funcionalidades do ConferenceXP não mencionadas neste trabalho como o *Venue Service* e o *Archive Service*, além da aplicação cliente do ConferenceXP não foram adaptadas para utilizar o CompactConferenceXP.

4.2. Informações de Contexto

Como a aplicação proposta deve ser sensível a contexto, esta deve ter acesso a informações como localidade e propriedades físicas dos dispositivos como energia restante da bateria, memória disponível, qualidade da conexão, entre

outras. Com estas informações, a aplicação poderá tomar ações automaticamente, sem a necessidade de interação direta com o usuário. Para o acesso a estas informações, foi utilizado um *middleware* chamado MoCA (Mobile Collaboration Architecture) [Sacramento, 2004], que oferece recursos para o desenvolvimento de aplicações sensíveis a contexto para computação móvel.

4.2.1. MoCA

Através de um conjunto de *APIs*, a MoCA provê serviços eficientes para a coleta, armazenamento e acesso a informações de contexto referentes a dispositivos móveis, e inferência e gerenciamento de informações sobre a localização geográfica de dispositivos. Desenvolvida na linguagem Java [Java, 2007], a MoCA não assume que a aplicação deva ser implementada de acordo com qualquer arquitetura específica, e seus serviços podem ser utilizados como base para o desenvolvimento de uma grande variedade de aplicações.

O monitor é o programa que executa em cada um dos dispositivos móveis, e é responsável por toda a coleta de dados a respeito do estado destes dispositivos. Todas estas informações são enviadas para um serviço da MoCA chamado CIS (Context Information Service). O CIS recebe, armazena e processa as informações de contexto enviadas por um conjunto de monitores. Este serviço também pode receber requisições de notificações de aplicações interessadas em estados específicos de determinados dispositivos. Quando um determinado dispositivo atinge determinado estado, o CIS dispara eventos notificando cada uma das aplicações interessadas.

A localização de um dispositivo é inferida pelo LIS (Location Inference Service) comparando os níveis dos sinais de rádio-freqüência recebidos dos pontos de acesso coletados pelo CIS, aos níveis previamente coletados em determinadas localidades [Nascimento, 2006].

A Figura 19 ilustra a arquitetura da MoCA, e exibe a comunicação entre as aplicações clientes que executam paralelamente os monitores, o CIS, o LIS, e os demais serviços da MoCA.

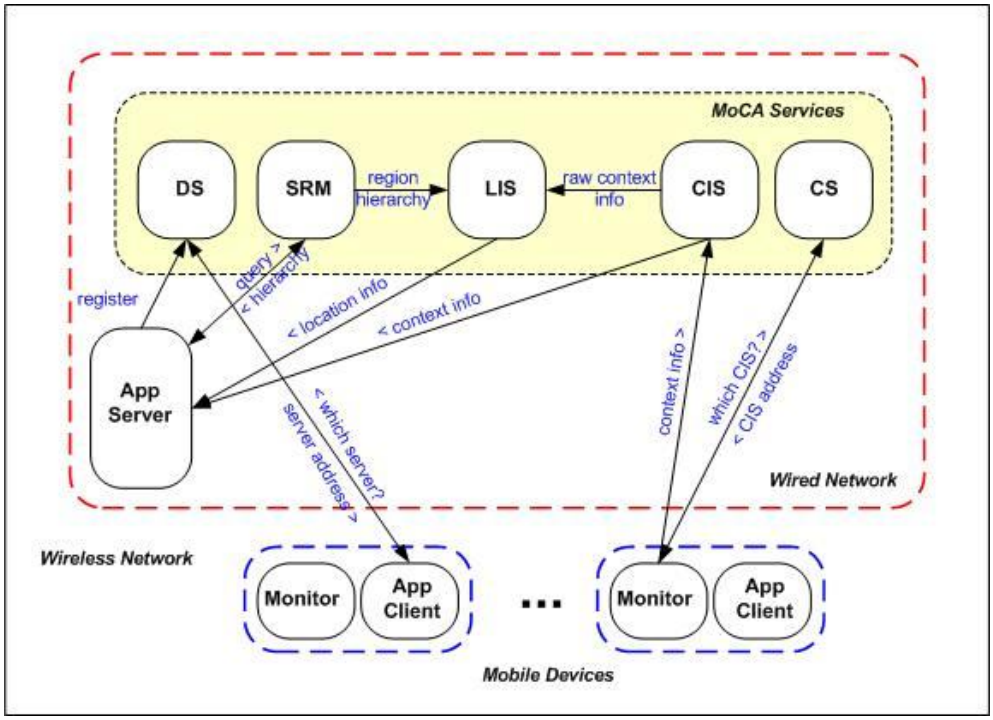

Figura 19 - A arquitetura da MoCA [Moca, 2007]

4.2.2. MoCA/WS

O MoCA/WS (MoCA/*Web Service*) [Malcher, 2006] é um *web service* que foi desenvolvido para que aplicações desenvolvidas em linguagens não-Java, como C++, Visual C#, Visual Basic, entre outras, possam acessar e utilizar os serviços de contexto oferecidos pela MoCA. Este *web service* atua como um cliente da MoCA, e através do envio e recebimento de mensagens SOAP (Simple Object Access Protocol) [SOAP, 2007] em requisições HTTP, qualquer aplicação pode acessar informações de contexto provenientes da MoCA.

O MoCA/WS atua como um *proxy* e oferece interface similar às próprias *APIs* da MoCA para aplicações Java. Quando uma aplicação cliente envia uma requisição ao MoCA/WS, este repassa a requisição à MoCA, que por sua vez retorna a resposta ao *web service.* A Figura 20 exibe a comunicação entre uma aplicação cliente, o MoCA/WS e a MoCA.

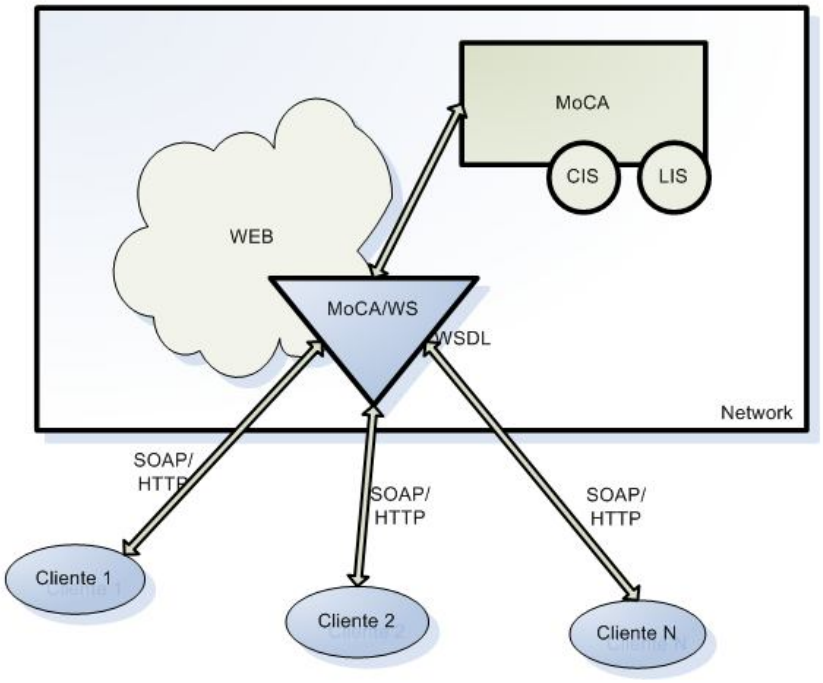

Figura 20 – MoCA x MoCA/WS x aplicações clientes [Malcher, 2006]

Como a aplicação proposta foi desenvolvida em Visual C#, o acesso às informações de contexto da MoCA foi realizado através do MoCA/WS.

5
A aplicação iPH

A aplicação desenvolvida neste trabalho é chamada de **Interactive Presenter for Handhelds – iPH** e permite o compartilhamento e a co-edição simultânea de uma apresentação colaborativa, é sensível a informações de contexto e pode ser executada em diferentes tipos de dispositivos como *tablet pcs*, *notebooks* e *handhelds*. Com o objetivo de ser utilizado em uma sala de aula, o iPH visa estimular uma participação ativa dos estudantes, aumentar o grau de aprendizado dos assuntos que foram ministrados, e prover uma rica experiência de uso através de uma interface simples e intuitiva e de auto-adaptações realizadas a partir de informações de contexto.

O iPH foi baseado no Classroom Presenter (vide seção 2.1) e possui funcionalidades semelhantes, como a forma de conexão e envio de mensagens, a utilização da caneta (*ink*) para realizar desenhos e enviá-los como contribuições ao participante mestre, entre outras. Por outro lado, algumas diferenças são notáveis como a possibilidade de inserir textos como contribuição, o acesso a informações de contexto e a criação de regras para adaptações dinâmicas e dependentes de contexto, bem como a adequação à execução em dispositivos portáteis (handhelds). Enquanto o Classroom Presenter foi desenvolvido para uso em *tablet pcs*, o iPH também pode ser executado em *handhelds* como *palmtops* e *smartphones*.

5.1.
Conceitos

Para melhor entender o funcionamento do iPH, alguns conceitos desenvolvidos para a aplicação devem ser detalhados. A Figura 21 exibe o modelo de entidade e relacionamentos do iPH.

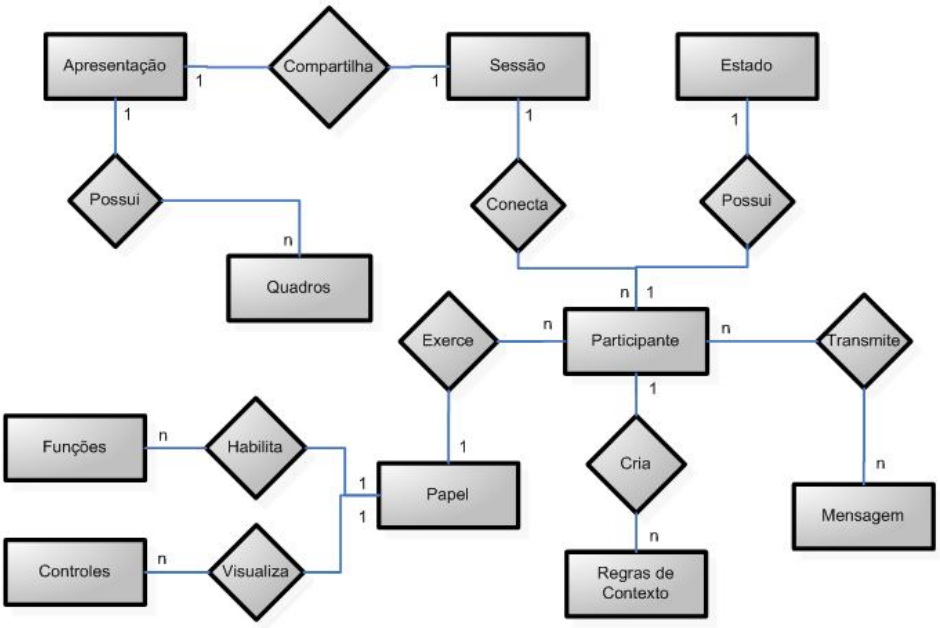

Figura 21 – Modelo de Entidades e Relacionamentos do iPH

5.1.1. Participantes

Um usuário se torna um participante quando conecta em uma sessão colaborativa, e possui os seguintes estados:

- **Conectado**: usuário está conectado em uma sessão colaborativa, sendo participante da mesma;
- **Desconectado**: usuário está desconectado, e, logo, não está participando de nenhuma sessão de colaboração.

Ao conectar-se em uma sessão colaborativa, o usuário deverá escolher qual papel irá exercer durante a mesma. Os papéis de contribuidor, mestre e visualizador são detalhados a seguir.

5.1.1.1. Contribuidor

O participante contribuidor (*contribuitor*) é como um estudante de uma aula, que acompanha a apresentação que está sendo compartilhada. Quando um

contribuidor se conecta ou desconecta de uma sessão, uma mensagem é enviada ao mestre informando que um contribuidor se conectou ou desconectou. O contribuidor pode fazer suas contribuições com *ink* e/ou texto e submeter estes quadros editados para o mestre.

Quando o mestre passa para um próximo quadro, todos os contribuidores que estiverem sincronizados com o mestre visualizarão o novo quadro. Esta sincronização pode ser quebrada de duas maneiras: quando o contribuidor explicitamente solicita a interrupção da sincronização com o mestre, ou quando o contribuidor começa fazer uma contribuição (e.g. selecionar a *ink* ou a entrada de um texto), o que o faz perder implicitamente a sincronização com o mestre, para que não seja interrompido durante a edição de sua contribuição. A qualquer momento, o contribuidor pode requisitar novamente esta sincronização.

O contribuidor pode acessar as suas informações de contexto e é o único dos participantes que está sujeito à adaptação a partir de regras de contexto criadas pelo mestre. De acordo com a configuração da aplicação, em um determinado período ocorre uma interpretação das regras criadas pelo mestre. Caso a condição de uma das regras seja verdadeira, a função associada será desabilitada.

5.1.1.2. Mestre

O participante mestre (*master)* é o participante que contém o maior número de funcionalidades no iPH. Como o professor de uma aula, o mestre decide qual apresentação será compartilhada entre os participantes da sessão colaborativa, e pode criar novos quadros públicos (i.e. *whiteboards*) que são visualizados pelos demais participantes, e quadros privados que são utilizados para anotações particulares.

O mestre possui o controle do andamento da apresentação. Quando o mestre visualiza um quadro, todos os participantes sincronizados visualizam também este quadro. As exceções são os quadros privados e quando os contribuidores não estão sincronizados com o mestre. Porém, o mestre pode solicitar aos contribuidores que se sincronizem novamente para acompanhar a apresentação. Em relação aos dispositivos no papel de visualizador, o mestre pode quebrar a sincronização com estes e assim, o quadro visualizado pelo mestre não será

atualizado para os visualizadores. A qualquer momento, o mestre poderá re-estabelecer esta sincronização. É válido ressaltar a diferença de sincronização entre os participantes: entre o mestre e contribuidor, quem possui o poder de quebrar e re-estabelecer a sincronização é o contribuidor, e o mestre pode apenas requisitar que os contribuidores sincronizem novamente; já entre o mestre e visualizador, quem controla a sincronização é o mestre.

Todas as contribuições realizadas pelo mestre são enviadas de maneira automática e instantânea para os participantes. Qualquer desenho ou texto é enviado aos participantes no momento em que eles são inseridos pelo mestre, e a essas contribuições estão associados os quadros em que foram realizadas. Desta maneira, evita-se que contribuições realizadas em um quadro, sejam desenhadas em outro quadro após o recebimento das mesmas. Sobre as contribuições dos outros participantes, o mestre tem a opção de configurar sua aplicação para que as aceite ou não.

Além de visualizar suas informações de contexto, o mestre visualiza também as informações de contexto dos outros participantes da sessão colaborativa. Desta maneira, o mestre tem uma visão geral dos estados dos dispositivos que estão sendo utilizados pelos participantes. Isto é importante, visto que o mestre pode criar regras de contexto para desabilitar as funções dos contribuidores, e obter um maior controle da apresentação colaborativa.

5.1.1.3. Visualizador

O participante visualizador (*viewer)* foi criado para auxiliar o mestre na exibição dos quadros da apresentação colaborativa e não realiza contribuições. A idéia do visualizador é que este esteja sendo executado em um dispositivo que esteja conectado a um projetor. O visualizador envia mensagens de conexão e desconexão para o mestre, e recebe todos os quadros da apresentação, exceto os quadros privados. Como o mestre, o visualizador recebe as contribuições dos contribuidores caso o mestre esteja as aceitando. Isto ocorre, pois o mestre pode desejar que uma contribuição de um participante seja exibida pelos visualizadores.

5.1.2. Interface

Como as versões do iPH foram desenvolvidas a partir do mesmo código, uma estrutura visual foi criada para que as versões apenas diferissem na implementação dos controles visuais, ou seja, na interface com o usuário. Portanto, determinadas áreas comuns a ambas as versões foram identificadas, e definidas para visualização de acordo com estado e papel do usuário na sessão. Os seguintes controles foram definidos:

1. **Painel de Funções Principais**: painel com as principais funções como a escolha do modo de contribuição;
2. **Painel de Funções Auxiliares**: painel com funções secundárias como navegação entre quadros;
3. **Painel de *Deck* Principal**: painel para visualização de todos os quadros da apresentação colaborativa;
4. **Painel de *Deck* de Contribuições**: painel para visualização de todas as contribuições enviadas pelos participantes;
5. **Painel de Visualização de Quadro:** painel para a visualização e edição de um quadro uma apresentação colaborativa.

Estes controles visuais são visualizados em ambas as versões, porém de uma forma diferente. Por exemplo, a versão iPH – XP disponibiliza o painel de visualização de um quadro simultaneamente com o painel de visualização de todos os quadros. Contudo, em *hanhdelds* isto não é possível, visto o *display* de tamanho limitado que possuem. Assim, a versão do iPH,- Mobile disponibiliza um painel de cada vez, com a opção de alternar entre os mesmos.

A Figura 22 e exibe a tela inicial do iPH-XP, identificando os controles visuais listados acima.

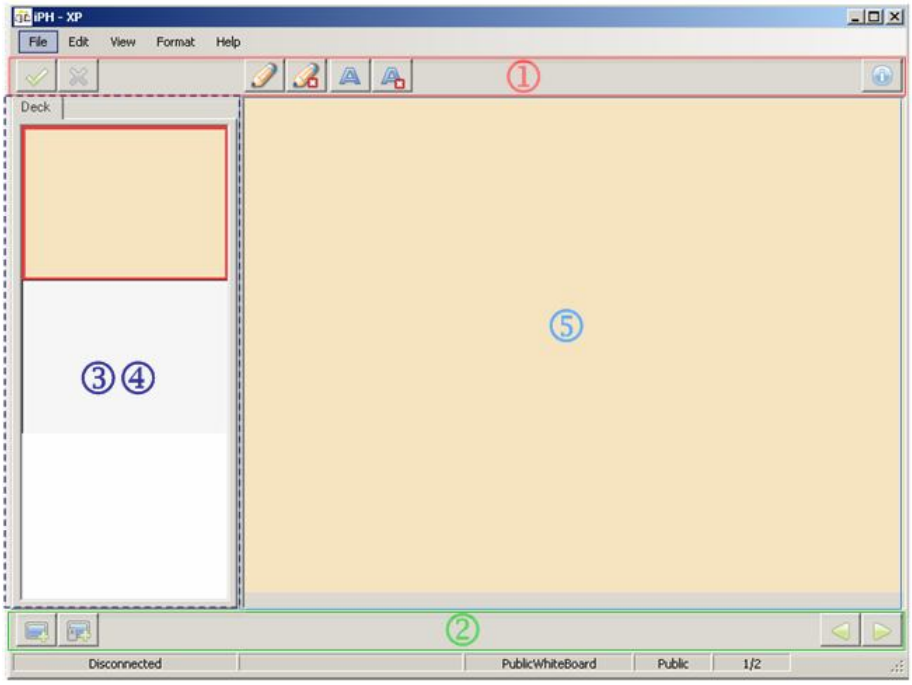

Figura 22 – Os controles visuais do iPH – XP

A versão iPH – Mobile apresenta os mesmos controles visuais, como exibido na Figura 23. A diferença é que estes controles não são visualizados simultaneamente.

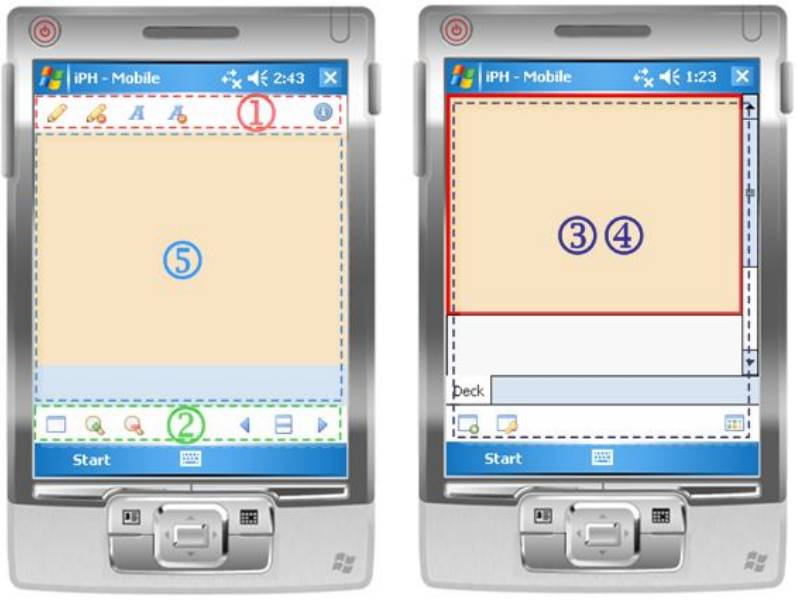

Figura 23 – Os controles visuais do iPH – Mobile

A Tabela 2 relaciona os painéis que são visualizados pelo usuário dependendo de seu estado e papel em uma sessão colaborativa.

Tabela 2 - Relação de painéis visualizados por cada participante

Painel	**Desconectado**	**Conectado**		
		Mestre	**Contribuidor**	**Visualizador**
Funções Principais	X	X	X	
Funções Auxiliares	X	X	X	
***Deck* Principal**	X	X	X	X
***Deck* de Contribuições**		X		
Visualização de Quadro	X	X	X	X

5.1.3. Sessão

O conceito de sessão no iPH está relacionado ao endereço de *multicast* utilizado para o envio e recebimento de mensagens e a senha utilizada nesta sessão. Isto se deve ao fato de que o iPH utiliza o CompactConferenceXP, que emprega uma aquitetura *peer-to-peer* (ponto-a-ponto). Apesar desta arquitetura facilitar a implantação da aplicação, ela tem a desvantagem de não possuir um repositório central de sessões. Como qualquer usuário que se conecte a determinado endereço *multicast*, poderá participar da sessão que estiver ocorrendo neste endereço, foi criado o conceito de senha da sessão. Todas as mensagens que são enviadas contêm a senha atribuída pelo usuário no momento da conexão, e estas mensagens somente serão processadas por usuários que se conectaram utilizando a mesma senha. Caso contrário, as mensagens serão descartadas.

Esta abordagem visa oferecer uma segurança a uma sessão colaborativa, fazendo com que somente os usuários que tenham o direito de participar, recebam as mensagens da sessão.

5.1.4. Conjunto de Quadros e Contribuições

Para o controle dos quadros e contribuições enviadas, foi necessário criar um modo de identificar unicamente estes objetos. Logo, cada quadro criado pelo participante mestre e enviado aos contribuidores e visualizadores possui uma identificação única, para assim evitar a duplicidade de quadros nestes participantes. Por exemplo, o mestre pode enviar o *deck* de quadros mais de uma vez. Somente os quadros que os demais participantes não possuírem serão inseridos em seus *decks*. Com este identificador, também é garantida a ordem dos quadros apresentados, pois, junto com o quadro enviado está o identificador do quadro anterior.

As contribuições estão relacionadas a um identificador de quadro, e elas também possuem um identificador único e um *timestamp* atribuído no momento da criação. Com o identificador do quadro, as contribuições enviadas são realizadas nos quadros corretos. Caso o participante receptor destas contribuições não possua o quadro identificado, estas contribuições são descartadas. É necessária a identificação para a opção de remover uma contribuição. Quando o mestre utiliza a borracha para remover contribuições, são enviados aos participantes os identificadores das contribuições que foram removidas. Os *timestamps* são utilizados para desenhar as contribuições na ordem correta.

5.1.5. Mensagens

As mensagens que devem ser enviadas e recebidas por um participante são definidas pelo papel que este exerce em uma sessão colaborativa. A seguir, a relação das mensagens, quais participantes enviam e recebem, e qual a ação tomada:

- **Conexão:** informa que um participante conectou em uma sessão de colaboração;

De	Para	Ação
Contribuidor	Mestre	Mestre insere contribuidor em lista de participantes
Visualizador	Mestre	Mestre insere visualizador em lista de participantes

- **Desconexão:** informa que um participante desconectou em uma sessão de colaboração;

De	Para	Ação
Contribuidor	Mestre	Mestre remove contribuidor em lista de participantes
Visualizador	Mestre	Mestre remove visualizador em lista de participantes

- **Ping:** requisita que os participantes enviem uma mensagem de conexão ao mestre;

De	Para	Ação
Mestre	Contribuidor	Contribuidor envia mensagem de conexão ao mestre
Mestre	Visualizador	Visualizador envia mensagem de conexão ao mestre

- **Adicionar quadro:** contém um quadro e as contribuições deste quadro (o envio de um *deck* é realizado enviando separadamente cada quadro);

De	Para	Ação
Mestre	Contribuidor	Contribuidor insere o quadro se não existir em seu *deck*, e insere as contribuições
Mestre	Visualizador	Visualizador insere o quadro se não existir em seu *deck*, e insere as contribuições

- **Quadro selecionado:** contém o identificador do quadro a ser selecionado do *deck* principal;

De	Para	Ação
Mestre	Contribuidor	Contribuidor seleciona o quadro se existir em seu *deck* e se estiver sincronizado com o mestre
Mestre	Visualizador	Visualizador seleciona o quadro se existir em seu *deck* principal e se estiver mensagem informar que deve selecionar

- **Requisição de atenção para quadro selecionado:** requisita sincronização para o quadro do *deck* principal identificado na mensagem;

De	Para	Ação

Mestre	Contribuidor	Contribuidor seleciona o quadro se existir em seu *deck* e se estiver sincronizado com o mestre. Caso não esteja sincronizado, contribuidor é avisado que o mestre solicitou sincronização

- **Quadro de contribuição selecionado:** requisita sincronização para o quadro do *deck* principal identificado na mensagem;

De	Para	Ação
Mestre	Visualizador	Visualizador seleciona o quadro se existir em seu *deck* de contribuições e se estiver mensagem informar que deve selecionar

- **Submissão de contribuição:** mensagem contendo o quadro editado com as contribuições de um participante contribuidor;

De	Para	Ação
Contribuidor	Mestre	Mestre insere o quadro no seu *deck* de contribuições se estiver aceitando submissões de participantes

- **Submissão de contribuição pelo mestre:** mensagem contendo o quadro editado com as contribuições de um participante contribuidor que foi aceito pelo mestre;

De	Para	Ação
Mestre	Visualizador	Visualizador insere o quadro no seu *deck* de contribuições

- **Adição de contribuição:** contém as contribuições associadas a um quadro de uma apresentação;

De	Para	Ação
Mestre	Contribuidor	Contribuidor insere contribuição no quadro associado se ainda não existir uma contribuição com o mesmo identificador
Mestre	Visualizador	Visualizador insere contribuição no quadro associado se ainda não existir uma contribuição com o mesmo identificador

- **Exclusão de contribuição:** contém as contribuições que devem ser removidas de um quadro de uma apresentação;

De	Para	Ação
Mestre	Contribuidor	Contribuidor remove as contribuições se existirem no quadro
Mestre	Visualizador	Visualizador remove as contribuições se existirem no quadro

- **Regras de Informação de contexto:** contém as regras de criadas para adaptar as funcionalidades dos participantes de acordo com suas informações de contexto;

De	Para	Ação
Mestre	Contribuidor	Contribuidor atualiza seu conjunto de regras com as regras enviadas

5.1.6. Acesso a Informações de Contexto e Regras de Contexto

Durante uma sessão de colaboração, todos os participantes podem acessar suas informações de contexto como localização, energia disponível, memória utilizada, entre outras. O mestre é o único que pode visualizar as informações dos outros participantes e criar regras a partir destas informações. Estas regras são compostas por uma condição a ser verificada por cada um dos participantes, e por uma funcionalidade associada, que será desabilitada caso a condição seja verdadeira. Por exemplo, o mestre pode decidir que todos os contribuidores que estiverem com um dispositivo com nível de energia menor que 10% não podem submeter contribuições. Quando o dispositivo de um contribuidor atingir um nível de energia menor que 10%, a aplicação desabilitará a opção de submissão de contribuições.

O mestre tem a opção de criar e remover uma ou mais regras e, periodicamente, estas regras são enviadas a todos os participantes contribuidores. De maneira automática e periódica, os contribuidores acessam o MoCA/WS, testam cada uma das condições presentes nas regras, e desabilitam as funcionalidades caso as condições sejam verdadeiras. A cada verificação, todas as funcionalidades são habilitadas novamente, pois uma condição previamente considerada verdadeira pode ter um resultado diferente.

5.2.
Exemplo de Uso do iPH

A utilização do iPH em uma sessão de colaboração é composta por várias etapas, iniciando na conexão do usuário e escolha do papel que este exercerá durante a sessão, até a visualização de todos os quadros da apresentação compartilhada, e conseqüentemente a desconexão. Durante este período, várias funcionalidades do iPH estão disponíveis como a sincronização entre participantes e o acesso a informações de contexto.

Para melhor explicar a utilização do iPH e suas funcionalidades, é utilizado um cenário de uma sala de aula composta por um professor e dois alunos. O professor exercerá o papel de mestre e irá comandar a apresentação a partir do seu *tablet pc*, e também utilizará um computador *desktop* conectado a um projetor fazendo o papel de visualizador. Os alunos conectarão como contribuidores e utilizarão *palmtops* para participar da aula e interagir com a apresentação compartilhada. A Figura 24 exibe estes participantes conectados através das tecnologias *ethernet* para o computador *desktop,* e *wi-fi* 802.11b para o *tablet pc* e os *palmtops.*

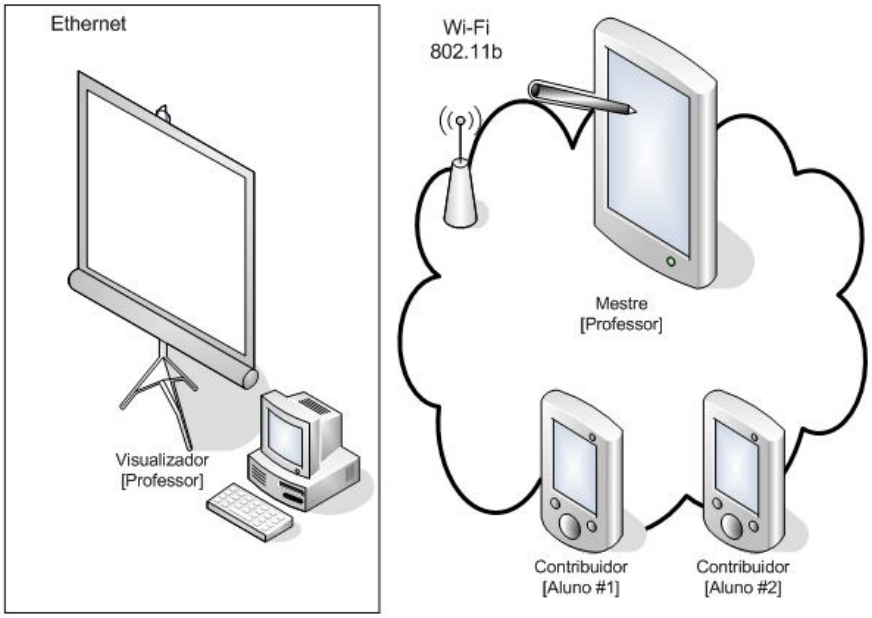

Figura 24 - O cenário de uso do iPH

5.2.1.
Conexão e Desconexão

A etapa inicial em uma sessão colaborativa é a conexão do usuário. Para conectar em uma sessão, os usuários devem entrar com os seguintes dados: o endereço IP de *multicast* e a porta utilizada para o envio e recebimento de mensagens; a senha da sessão colaborativa; o nome do usuário dentro da sessão; e o papel que o usuário exercerá na sessão. Também é utilizado o endereço MAC das placas de rede, porém este dado é coletado automaticamente pela aplicação.

Para abrir o formulário de conexão, o usuário do iPH deve escolher o item *File* do *menu*, e clicar em *Connect*. Ao conectar o usuário poderá transmitir mensagens pelo endereço de *multicast*. A Figura 25 exibe o formulário de conexão para as versões do iPH.

Figura 25 - O formulário de conexão do iPH

O professor utilizando o *tablet pc* escolherá o papel de mestre (*master*) na aba *User*, e para o computador conectado ao projetor escolherá o papel de visualizador (*viewer*). Os dois alunos escolherão o papel de contribuidor (*contribuitor*) para a conexão[3].

[3] Nesta primeira versão do iPH, não houve a preocupação com o controle de qual papel cada usuário poderá exercer durante uma sessão colaborativa.

5.2.2.
Visualização de Quadros

Após efetuar a conexão em uma sessão de colaboração, o iPH disponibiliza funcionalidades de acordo com o papel que o usuário exercerá na sessão. O mestre é o único que pode inserir quadros públicos e privados durante a sessão, e possui opções de envio de quadros aos outros participantes. Os contribuidores e visualizadores recebem estes quadros, porém, somente os contribuidores podem navegar entre eles. Assim como o mestre, os contribuidores possuem opções de avanço e de retrocesso na apresentação e possuem um painel para visualização de todos os quadros da apresentação.

A Figura 26 exibe as interfaces para os diferentes usuários: o professor fazendo o papel de mestre; os alunos conectados como contribuidores; e o computador conectado ao projetor exercendo papel de visualizador. É possível notar a diferença dos componentes visuais disponíveis de acordo com a seção 5.1.2.

Figura 26 - Os usuários conectados com diferentes papéis

5.2.3.
Modos de Contribuição

Para contribuir com um quadro da apresentação, o mestre e os contribuidores têm as opções de *ink* e texto, e a borracha para cada uma delas. Os visualizadores não podem contribuir, e, portanto, não visualizam os controles de contribuição. A Figura 27 exibe os controles disponíveis para o mestre e contribuidores.

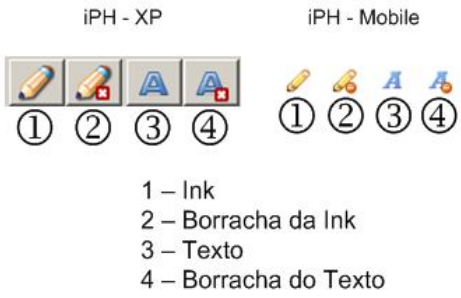

Figura 27 - As opções de contribuição do iPH

Quando um participante escolhe o modo de contribuição Texto, ao clicar em qualquer ponto da área de visualização de um quadro, uma tela solicitando o texto a ser inserido é visualizada. Após digitar o texto desejado, o mesmo é inserido como uma contribuição ao quadro atual. O participante também pode formatar a cor da *ink* e a cor e fonte do texto a ser inserido.

5.2.4.
Distribuição da apresentação e contribuições

A distribuição da apresentação é realizada pelo mestre que envia toda a apresentação ou um único quadro da apresentação, incluindo as contribuições realizadas em cada quadro. Os demais participantes recebem estes quadros e inserem em seus *decks*. Toda contribuição realizada pelo mestre é imediatamente enviada aos demais participantes. Por sua vez, um contribuidor pode realizar suas contribuições e em determinado momento, enviar o conjunto das mesmas para o mestre. Na Figura 28, o professor envia uma apresentação para os demais participantes e todas as contribuições realizadas pelo professor são imediatamente visualizadas pelo visualizador e pelos contribuidores.

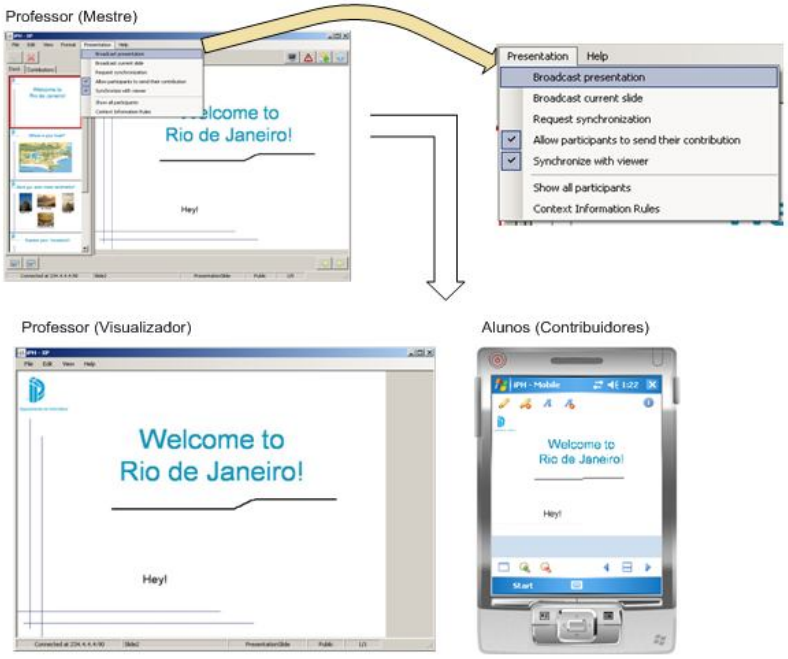

Figura 28 - O envio de uma apresentação para os participantes

O mestre pode decidir não receber mais contribuições dos contribuidores. Para isto, basta desmarcar a opção de *menu* ***Allow participants to send their contribution*** e as novas contribuições recebidas serão rejeitadas. Na versão iPH – XP, há também um botão que indica esta decisão, como na Figura 29.

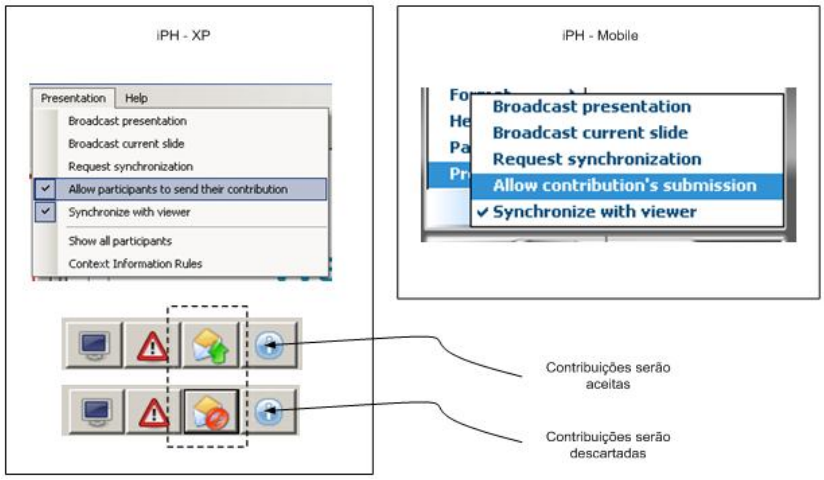

Figura 29 - Participante mestre pode aceitar ou descartar contribuições

Para submeter uma contribuição ao mestre, basta que o contribuidor clique na opção de *menu* ***Send Contribution*** para enviar o quadro atual, que será

recebido pelo mestre e inserido no *deck* de contribuições do mesmo, caso esteja aceitando contribuições. A Figura 30 exibe uma contribuição submetida por um aluno, e a mesma visualizada pelo professor.

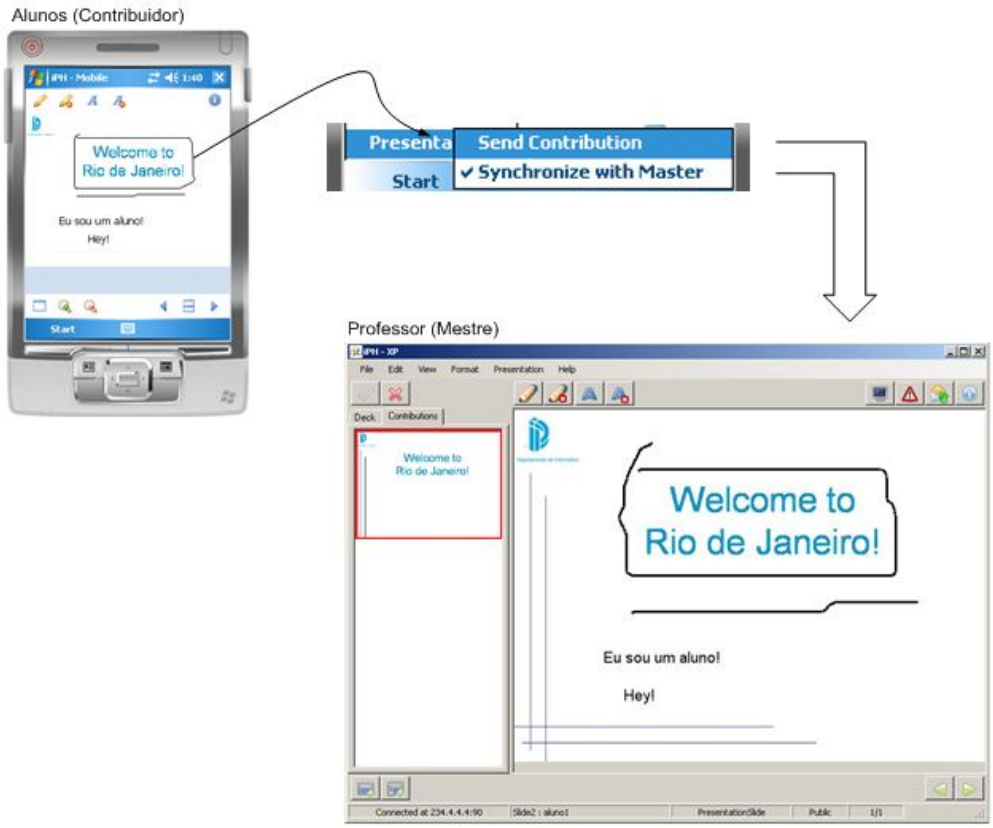

Figura 30 - A submissão de uma contribuição ao mestre

5.2.5. Sincronização entre participantes

A sincronização dos quadros de uma apresentação está relacionada aos papéis exercidos pelos participantes. O mestre possui controle do que é visualizado em um visualizador, o que não acontece em relação aos contribuidores, que podem navegar entre os quadros sem a necessidade de estarem sincronizados com o mestre.

Após a conexão e o recebimento da apresentação, todos os participantes estão sincronizados com o mestre. Quando o mestre navega para um próximo quadro, todos os participantes visualizam este quadro. A exceção é quando o

quadro é privado, que não chega a ser enviado para os demais participantes. Se o mestre desejar que o próximo quadro não seja visualizado pelos visualizadores, ele deve desmarcar a opção de *menu* ***Synchronize with viewer***, como na Figura 31.

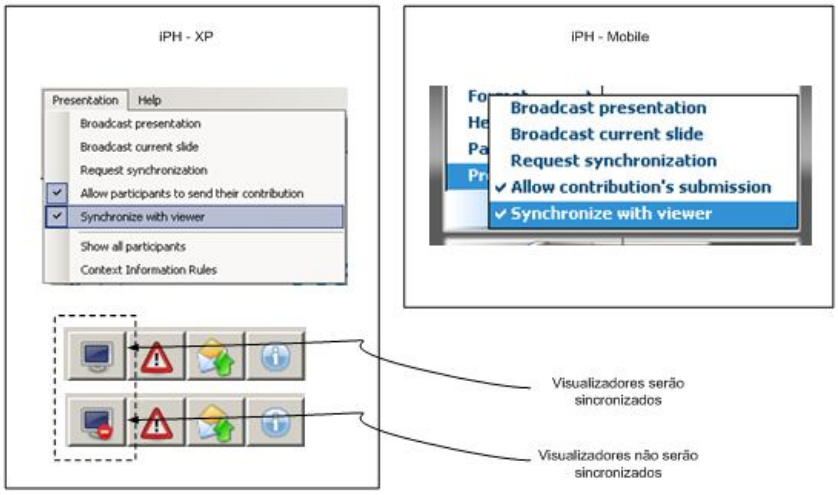

Figura 31 - Sincronização dos visualizadores

Os contribuidores perdem a sincronização com o mestre quando efetuam uma contribuição ou quando explicitamente solicitam isto ao desmarcar a opção de *menu* ***Synchronize with Master***. Este mesma opção de *menu* deve ser marcada quando o contribuidor desejar sincronizar novamente com o mestre. A Figura 32 exibe estas opções.

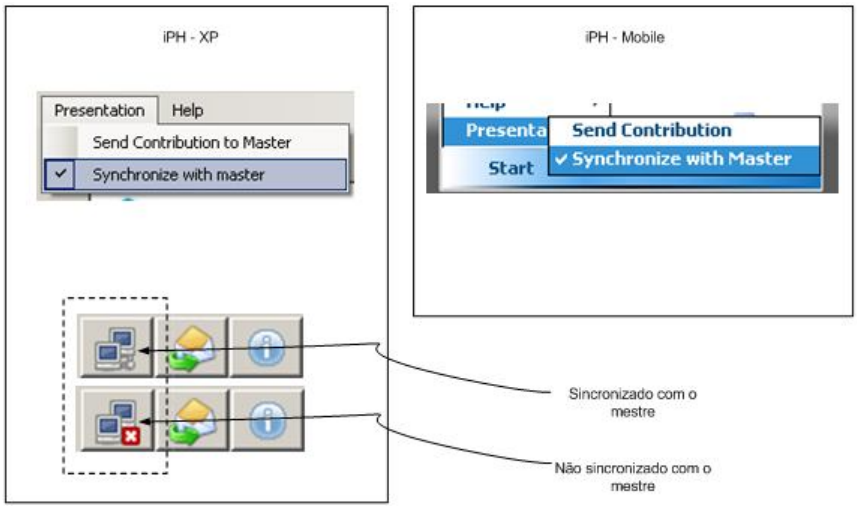

Figura 32 - Sincronização dos contribuidores com o mestre

Apesar de o mestre não ter controle da sincronização dos contribuidores, ele pode solicitar que esta sincronização seja re-estabelecida clicando na opção de *menu* ***Request Synchronization***. Assim, os contribuidores são avisados de que o mestre está requisitando sincronização.

5.2.6. Informação de Contexto: Acesso e Regras

Todos os participantes podem visualizar suas próprias informações de contexto clicando na opção de *menu* ***Context Information***. A Figura 33 exibe o formulário com as informações de contexto do dispositivo, no caso da conexão com o MoCA/WS ter sido estabelecida. Caso não seja possível acessar o *web service*, o usuário é alertado de que as informações de contexto estão indisponíveis.

Figura 33 - O acesso a informações de contexto

Para acessar a lista de participantes da sessão de colaboração e suas informações de contexto, o mestre deve clicar na opção de *menu* ***Show all participants*** como na Figura 34. Um formulário com a lista de todos os participantes e suas informações de contexto é exibido diferenciando com um ícone os contribuidores e visualizadores.

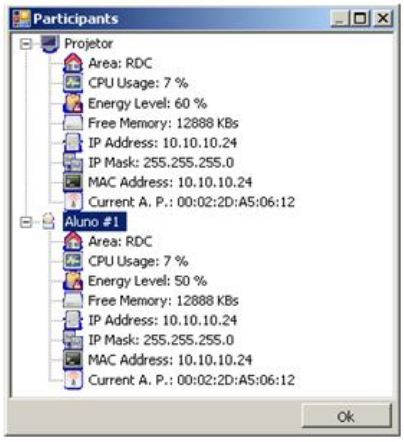

Figura 34 - Listagem de participantes e suas informações de contexto

O mestre possui também a opção de *menu* ***Context Information Rules*** para criar e remover as regras de informação de contexto da sessão de colaboração. Para criar uma regra, o mestre deve definir uma condição a ser avaliada e a funcionalidade associada. A Figura 35 exibe os formulários de listagem de regras e criação de uma nova regra.

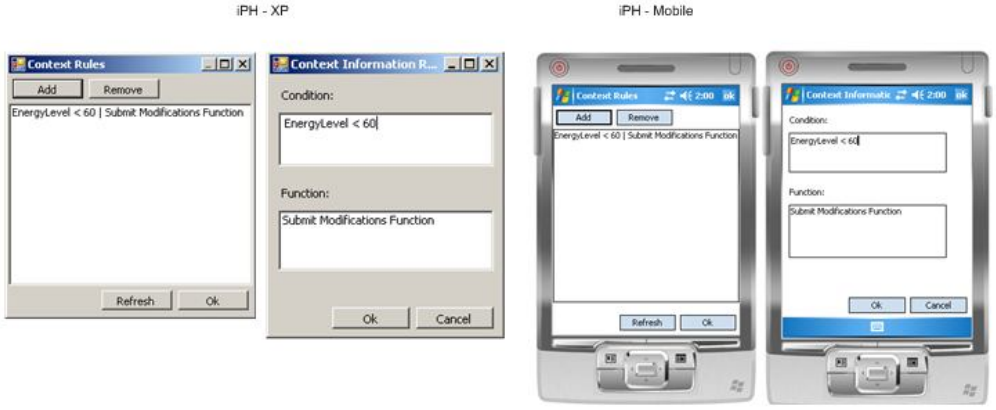

Figura 35 - As regras de informação de contexto

Caso a condição de uma regra de contexto seja verdadeira, a funcionalidade associada a esta é desabilitada, o que pode causar também modificações na interface. A Figura 36 ilustra um exemplo de uma regra aplicada nos participantes contribuidores, no qual a função de submissão de contribuições ao mestre é desabilitada devido a uma regra de contexto. Os contribuidores ficam impossibilitados de submeter suas contribuições até que a regra seja retirada pelo mestre ou o resultado de sua condição seja falso.

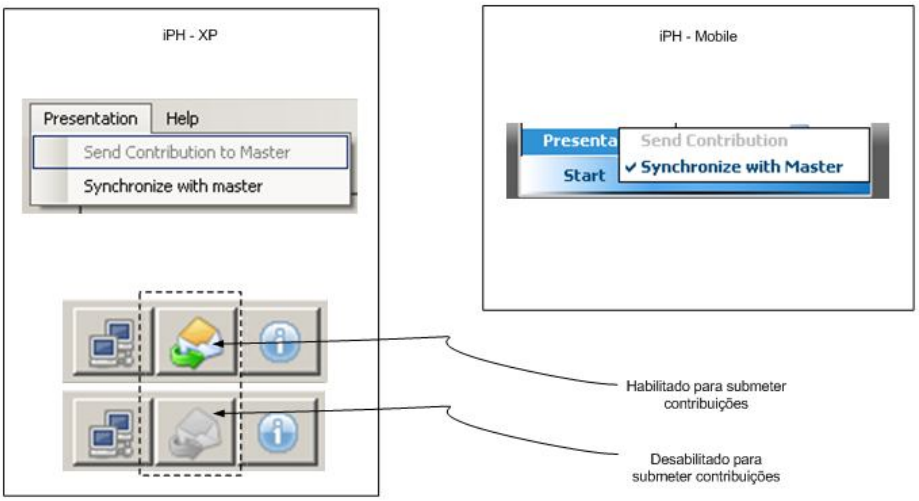

Figura 36 - Funcionalidade desativada por uma regra de contexto

5.2.7. Configuração da aplicação

Durante o uso do iPH, todos os participantes tem a opção de configurar sua aplicação acessando o formulário de configuração através da opção de *menu* ***Configuration***. Neste formulário, o usuário poderá alterar o endereço do MoCA/WS utilizado para acessar as informações de contexto, a opção de utilizar ou não as regras de contexto para adaptação, atribuir o período em que as condições das regras de contexto são testadas, e o tempo limite de conexão com o MoCA/WS. Além destas informações, o usuário também poderá escolher as cores dos quadros públicos e privados que vierem a ser inseridos. A Figura 37 exibe este formulário.

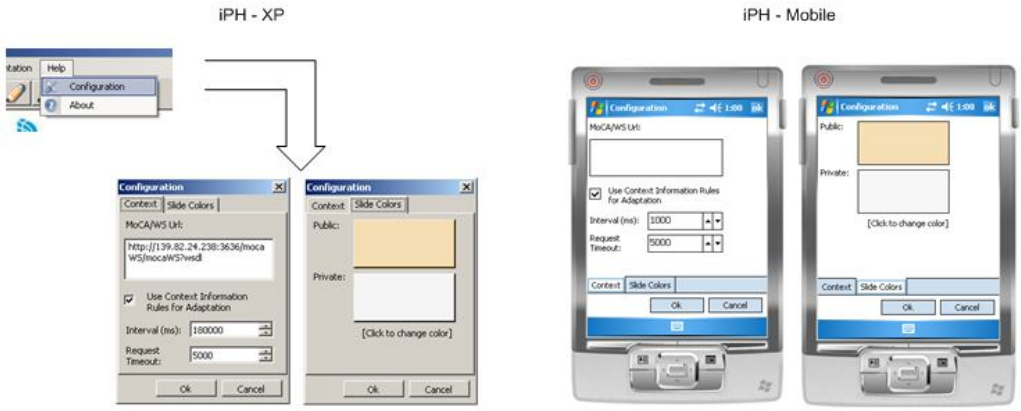

Figura 37 - Formulário de configuração do iPH

6
Implementação do iPH

O **iPH** foi desenvolvido utilizando a linguagem Visual C# da plataforma .NET[4] e possui duas versões: uma versão para dispositivos que executem o Windows XP e o .NET Framework, chamada **iPH - XP**; e outra para ser executada em dispositivos computacionais mais limitados como *palmtops* e *smartphones*, executando o Windows Mobile e .NET Compact Framework (.NET CF). Esta última versão é chamada **iPH - Mobile**. A Tabela 3 exibe os requisitos de execução das versões do iPH.

Tabela 3 - Versões do Interactive Presenter for Handhelds - iPH

Versão	Dispositivos Computacionais	Sistema Operacional[5]	Plataforma .NET
iPH – XP	*Desktops, notebooks* e *tablet pcs*	Windows XP	.NET Framework 2.0
iPH – Mobile	*Palmtops* e *smartphones*	Windows Mobile	.NET Compact Framework 2.0

Qualquer uma das versões pode ser executada localmente, sem necessidade de interface de rede e/ou conexão em uma sessão colaborativa. Funcionalidades como a visualização de um *deck* e a edição dos quadros dos mesmos estão disponíveis. Ao utilizar o iPH de maneira distribuída, conectado em uma sessão de colaboração, pode-se constatar suas demais funcionalidades como: o envio e recebimento de quadros e contribuições entre os participantes; a submissão de

[4] Para o desenvolvimento do iPH - XP e do iPH - Mobile foram utilizadas as versões 2.0 da .NET Framework e .NET Compact Framework.

[5] Apesar de não terem sido feitos testes utilizando outros sistemas operacionais, como Windows 2000 e Windows Vista para o iPH - XP e Windows CE 4 para iPH - Mobile, acredita-se que estas aplicações podem ser executadas normalmente, bastando que esteja instalado a plataforma .NET correspondente.

contribuições; o controle de sincronia entre os diferentes participantes; o gerenciamento de regras de contexto; entre outras.

Apesar das versões do iPH serem executadas em plataformas diferentes, elas compartilham praticamente o mesmo código-fonte, variando apenas em detalhes de interface com o usuário e acesso a arquivos de configuração. Isto é possível, pois, como mencionado, praticamente todo código desenvolvido para o .NET CF é compatível com o .NET Framework. Estas funcionalidades do iPH foram divididas em diversos pacotes que são utilizados por ambas as versões. Apesar de esta decisão ter aumentado a complexidade do código, visto que existem funcionalidades disponíveis apenas para o .NET Framework, ela agilizou o desenvolvimento do aplicativo, pois não houve necessidade de re-trabalho para desenvolver uma mesma funcionalidade para diferentes plataformas. Isto também facilitou o gerenciamento do desenvolvimento e a aplicação de testes.

O desenvolvimento do iPH foi realizado em três etapas:

1. Desenvolvimento de funcionalidades básicas necessárias ao iPH e que pudessem ser re-aproveitadas por qualquer outro projeto .NET;
2. Desenvolvimento de funcionalidades específicas ao iPH e comum a ambas as plataformas .NET;
3. Desenvolvimento da interface do usuário e funcionalidades específicas de cada versão do IPH.

As funcionalidades mencionadas acima foram divididas em pacotes que foram implementados através de bibliotecas compatíveis com o .NET CF.

6.1.1. Funcionalidades Básicas - Pacote LAC

Durante o desenvolvimento do iPH, foram identificadas várias funcionalidades básicas que poderiam ser reaproveitadas em quaisquer outros projetos. Foi criado o pacote **LAC**, que possui funcionalidades de comunicação, acesso a informações de contexto e funções gerais como: redimensionamento de imagem, formulários de entrada de dados e outras. O pacote LAC está dividido em três pacotes, como exibido na Figura 38.

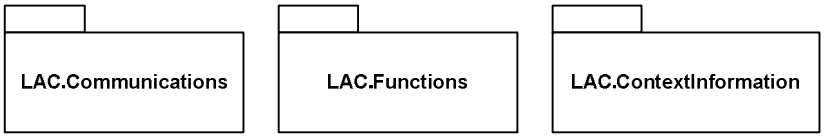

Figura 38 - O pacote LAC

Esses três pacotes são utilizados e distribuídos de maneira independente, sendo que cada um deles foi implementado como uma biblioteca do .NET CF. A Tabela 4 resume os pacotes que são detalhados a seguir.

Tabela 4 - Descrição do pacote LAC

Pacote	Descrição
LAC.Communications	Contém classes que fazem a comunicação entre aplicações distribuídas através do CompactConferenceXP. É através destas classes que são enviadas e recebidas as diversas mensagens durante uma sessão de colaboração do iPH.
LAC.ContextInformation	O pacote contém classes que encapsulam chamadas ao MoCA/WS, facilitando o acesso e o uso de informações de contexto.
LAC.Functions	Controles e classes que não são específicos da aplicação foram desenvolvidos neste pacote. Funcionalidades como formulários de entrada de dados, funções de redimensionamento de imagens, controles visuais inexistentes no .NET CF, entre outras.

6.1.1.1. LAC.Communications

O pacote LAC.Communications contém classes que fazem a comunicação entre as aplicações durante uma sessão de colaboração, encapsulando todas as chamadas ao CompactConferenceXP. A Figura 39 exibe as classes e a interface que pertencem a este pacote.

Figura 39 - O pacote LAC.Communications

Para que uma aplicação possa enviar e receber mensagens durante uma sessão colaborativa, basta que ela possua um atributo da classe *CommunicationControl*, e passe como argumento do método construtor deste, um objeto de uma classe que implemente a interface *ICommunications*. Esta interface possui um só método a ser implementado, chamado *ReceiveObject*, que é utilizado para processar as mensagens recebidas.

A classe *CommunicationControl* oferece métodos para conexão e desconexão, e possui um atributo da classe *MessageSender* utilizado para envio de mensagens pela rede através do método *SendObject*, que insere a mensagem em uma fila de mensagens a serem enviadas pertencentes à classe *SendingQueue.* Esta classe possui uma *thread* que retira as mensagens da fila e as envia através de chamadas a API do CompactConferenceXP. Quando a fila está vazia, esta *thread* permanece em modo de espera, até a próxima mensagem ser inserida na fila.

A classe *Utility* possui um método estático que serializa as mensagens utilizando o CompactFormatter antes de inserir na fila de envio. Quando a mensagem é recebida, esta é deserializada pelo objeto *CommunicationControl*.

A Figura 40 exibe a comunicação entre diferentes versões do iPH. É válido lembrar que este pacote é independente, e logo, qualquer aplicação pode fazer uso do mesmo para comunicação através do CompactConferenceXP.

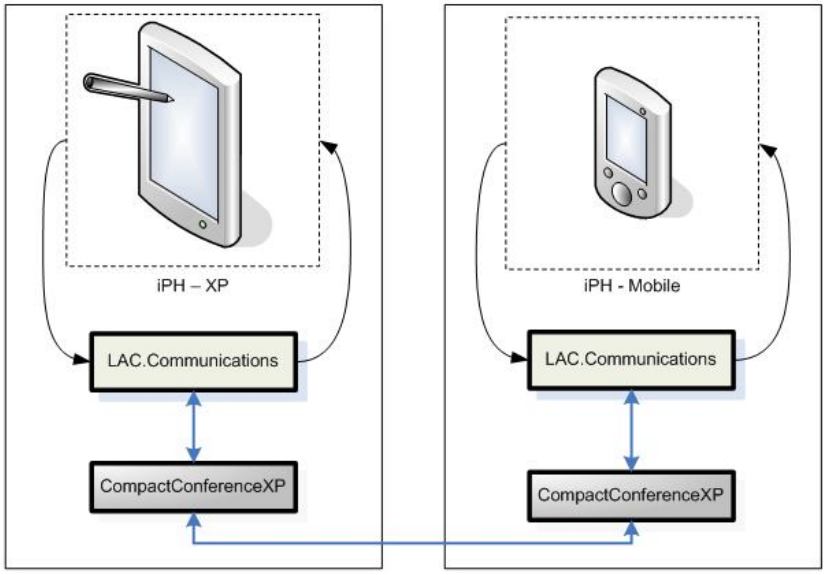

Figura 40 - Comunicação através do LAC.Communications

O pacote LAC.Communications foi baseado na implementação da comunicação do aplicativo ClassroomPresenter, que possui uma maneira semelhante de enviar e receber mensagens utilizando o ConferenceXP.

6.1.1.2. LAC.ContextInformation

Este pacote foi criado com o objetivo de facilitar o acesso a informações de contexto por qualquer aplicação .NET. Além da própria referência ao MoCA/WS (vide seção 4.2.2), existem duas classes neste pacote: a classe *ContextWrapper*, que contém os métodos para o acesso a informações de contexto; e a classe *DeviceContext*, que encapsula informações de contexto de um determinado dispositivo. A Figura 41 exibe as classes mencionadas e os métodos utilizados para acessar as informações de contexto fornecidas pelo MoCA.

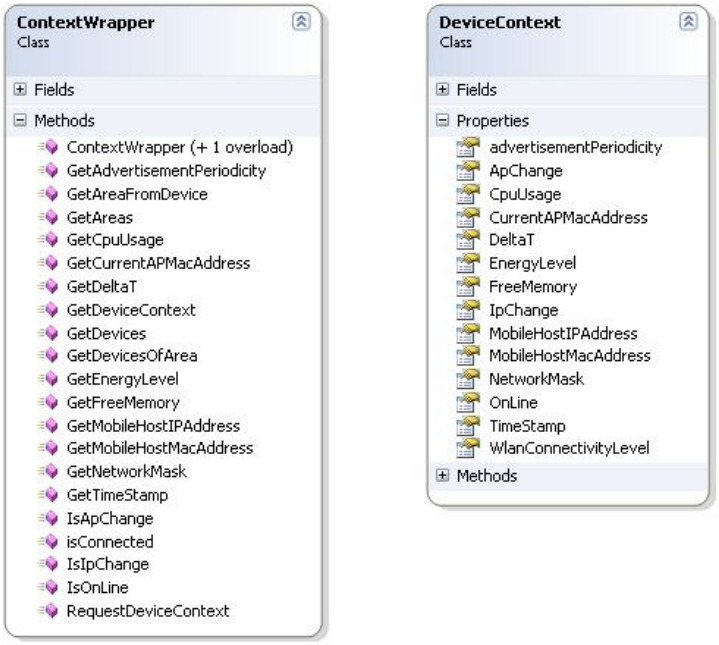

Figura 41 - As classes do pacote LAC.ContextInformation

Para fazer uso do LAC.ContextInformation, uma aplicação deve possuir um atributo do tipo *ContextWrapper*, e passar como argumento do seu método construtor o endereço do MoCA/WS. Com isto, a aplicação pode fazer requisições a respeito de informações de contexto como: energia disponível; qualidade do enlace; localização simbólica de um dispositivo; dispositivos presentes em uma localização simbólica; e outras.

A Figura 42 exibe o acesso a informações de contexto através do LAC.ContextInformation, que encapsula a referência ao MoCA/WS e suas chamadas.

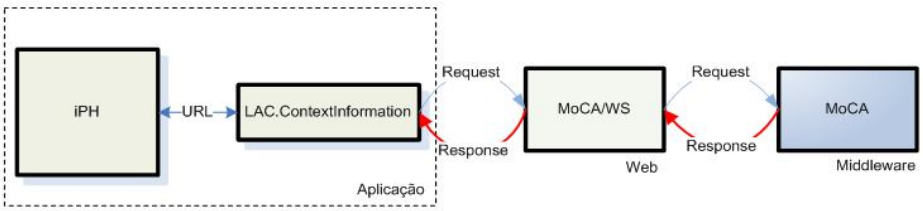

Figura 42 – Utilização do LAC.Communications

6.1.1.3. LAC.Functions

O pacote LAC.Functions é bastante simples e objetiva oferecer funcionalidades básicas que não possuem uma categoria específica como comunicação ou informação de contexto. Este pacote é subdivido nos pacotes: *Drawing*, *Forms*, *Networks* e *Objects*.

O sub-pacote *Drawing* contém: uma classe chamada *DrawingFunctions*, que oferece métodos estáticos relacionados ao tratamento de imagens; o método *CreateImageFromImage*, que retorna uma imagem redimensionada a partir de outra imagem, passando como argumentos o tamanho da nova imagem e o tamanho da imagem original na nova imagem, utilizado pelo iPH para o desenho dos quadros de uma apresentação; e o método *GetRelativeSize*, que retorna a altura e largura relativas a uma relação original passada como parâmetro. Este último é utilizado no desenho dos quadros de uma apresentação para desenhar os e mesmos sempre em uma mesma proporção.

Dentro do pacote *Forms* encontram-se três tipos de formulários: *AboutBox*, utilizado para prover informações sobre a aplicação que está sendo executada; *InputBox*, que é utilizado para requisitar ao usuário a entrada de texto; e *InputTrackBarForm*, utilizado para requisitar ao usuário que escolha um valor inteiro dentro de uma margem pré-estabelecida. Somente os dois primeiros formulários foram utilizados para o iPH, onde o *AboutBox* é o formulário utilizado para fornecer as informações da aplicação, e o *InputBox* é utilizado durante vários momentos como a configuração do nome de um quadro adicionado e a adição de um texto em um quadro. A Figura 43 exibe o formulário *InputBox* em ambas as plataformas .NET.

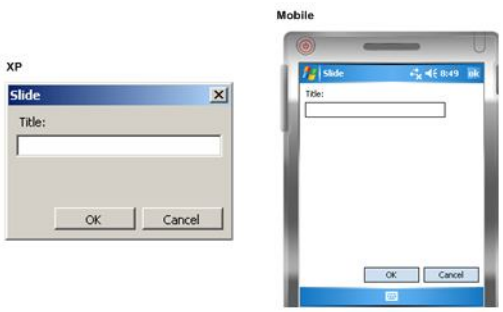

Figura 43 - *InputBox* sendo visualizado em ambas as plataformas

O pacote *Networks* contém uma única classe chamada *PhysicalAddress*, que possui um método estático que retorna o endereço MAC do dispositivo computacional que está executando a aplicação. Este endereço é necessário para fazer as chamadas ao MoCA/WS, e assim recuperar as informações de contexto do dispositivo utilizado. Por último, o pacote *Controls* contém uma única classe chamada *ImageButton*, que representa um botão associado a uma imagem e que foi criada para a versão iPH – Mobile.

O objetivo do pacote LAC.Functions é oferecer funcionalidades comuns a aplicações desenvolvidas para plataforma .NET, e à medida que este pacote seja utilizado por outros projetos, outras funcionalidades serão necessárias e inseridas no pacote, tornando-o cada vez mais útil.

6.1.2. Funcionalidades Específicas – Pacote iPH

O pacote **iPH** contém as classes e interfaces específicas ao aplicativo desenvolvido, como as classes que representam uma apresentação e seus quadros, as classes das mensagens transmitidas, as classes de formulários de interface com o usuário, entre outras, além de fazer uso das funcionalidades oferecidas pelo pacote LAC. Este pacote está dividido em três pacotes que se relacionam como na Figura 44.

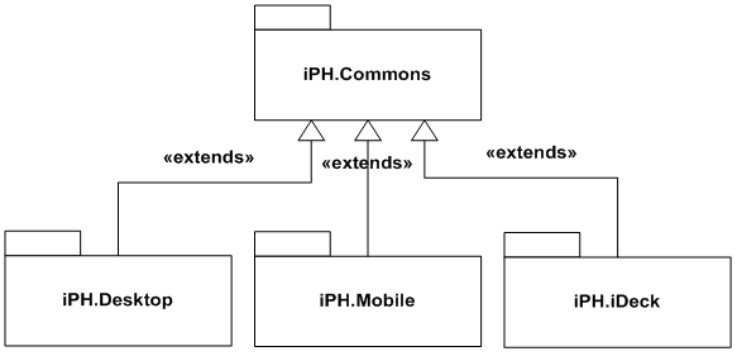

Figura 44 - O pacote iPH

O pacote iPH.Commons contém todas as classes que são utilizadas por todas as versões, enquanto que os pacotes iPH.Desktop e iPH.Mobile possuem classes específicas a versão do iPH utilizada. O pacote iPH.iDeck contém as classes relacionadas ao aplicativo iDeck, utilizado para criação de arquivos de

decks, e é descrito no anexo 10.1. A Tabela 5 contém um breve resumo desses pacotes, que são detalhados a seguir.

Tabela 5 - Descrição do pacote iPH

Pacote	Descrição	Plataforma
iPH.Commons	Contém diversas classes e interfaces necessárias para o desenvolvimento do iPH como: classes que representam os quadros de uma apresentação, classes de mensagens; formulários utilizados em ambas as versões, classes de controles visuais específicos, gerenciadores de participantes, regras de contexto, entre outras.	.NET CF
iPH.Desktop	Contém as classes que implementam a interface visual do iPH – XP, e algumas funcionalidades específicas para a plataforma .NET Framework.	.NET Framework
iPH.Mobile	Contém as classes que implementam a interface visual do iPH – Mobile, e algumas funcionalidades específicas para a plataforma .NET CF.	.NET CF
iPH.iDeck	Contém as classes pertencentes ao aplicativo iDeck, utilizado para criação de *decks*. Este aplicativo é descrito no Anexo 10.1.	.NET Framework

6.1.2.1. iPH.Commons

O pacote iPH.Commons é o núcleo do iPH, pois nele estão contidas as classes e interfaces que são utilizadas por ambas as versões da aplicação. Para executar funcionalidades específicas, essas classes disponibilizam métodos virtuais e abstratos que são implementados por cada uma das versões. Este pacote está subdividido em diversos outros pacotes que relacionam diversas áreas do iPH.

O pacote **Configuration** contém a classe *InteractiveConfiguration*, que define a configuração da aplicação com atributos como: o endereço do MoCA/WS utilizado; as cores para quadros públicos e particulares; a utilização de regras de contexto para adaptação; e o período em que estas regras são aplicadas. Esta classe possui valores padrão, porém seus atributos podem ser configurados por arquivos de configuração ou durante a execução da aplicação.

O pacote **Context** contém a classe que representa uma regra de contexto, chamada *ContextInformationRule,* e as classes que aplicam estas regras durante uma apresentação, dependendo do papel que está sendo exercido pelo participante. Para isto foi utilizado o padrão *Factory*[6]. A Figura 45 exibe as classes deste sub-pacote.

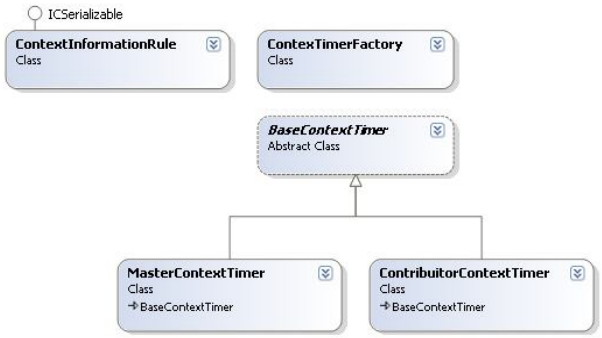

Figura 45 - As classes do sub-pacote iPH.Commons.Context

Em **Forms** estão contidas todas as classes de formulários utilizados por ambas as versões do iPH, que são: formulário de configuração, utilizado para configurar a aplicação; formulário de sessão, utilizado para ingressar em uma sessão de colaboração; formulário de informações de contexto do participante; formulário de listagem e adição de regras de contexto; formulário de listagem de participantes de uma sessão colaborativa; e o formulário principal da aplicação iPH, o *InteractivePresentationForm*. Esta última classe contém as principais funcionalidades do iPH, é abstrata e deve ser implementada pelas versões específicas de cada plataforma.

O pacote **Functions** contém as classes que representam as funções disponíveis no iPH, e são utilizadas para definir o que cada tipo de participante poderá executar e visualizar na aplicação. Essas classes também são utilizadas na definição de regras de contexto, pois uma regra é definida por uma condição e a função que desabilitará se esta condição for verdadeira.

---

[6] *Factory* é um padrão de projeto que permite a criação de objetos ou famílias de objetos relacionados ou dependentes, através de uma única interface e sem que a classe concreta seja especificada [Gamma, 1995].

O pacote **Ink** possui uma única classe contendo um único método estático utilizado para definir a largura da *ink*. Dependendo do sistema operacional que está sendo utilizado, Windows Mobile ou Windows XP, a largura das linhas desenhadas difere, visto que para um *handheld* esta largura deve ser menor que a largura de uma linha desenhada em um *notebook*.

No pacote **Manager** estão todos os gerenciadores utilizados pela aplicação. Esses gerenciadores controlam as regras de contexto existentes, a lista de participantes conectados em uma sessão de colaboração, as contribuições de cada quadro de uma apresentação colaborativa, entre outros.

As classes de mensagens que são transmitidas entre as aplicações do iPH estão contidas no pacote **Messages.** A classe base é chamada de *BaseMessage* e possui como atributos a chave da sessão de colaboração, um *timestamp* da criação desta mensagem, e o participante remetente. As outras classes derivam desta classe base como a classe *DeckMessage*, que representa um *deck*, e a classe *SlideAddMessage*, que contém um quadro a ser inserido pelos participantes que recebem esta mensagem. Todas as classes deste pacote implementam os métodos definidos pela classe *ICSerializable* do CompactFormatter para serialização e deserialização. A Figura 46 exibe as classes do pacote Messages:

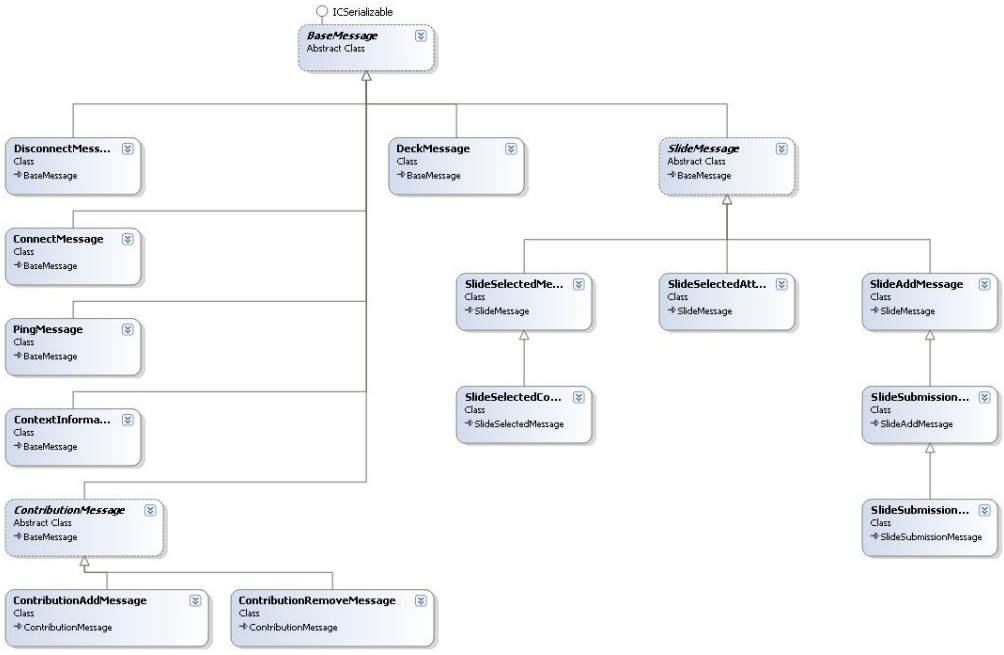

Figura 46 - As classes do sub-pacote iPH.Commons.Messages

O pacote **Parameters** contém uma classe que disponibiliza os parâmetros utilizados durante uma sessão colaborativa, como a definição de que um participante pode ou não enviar contribuições, o sincronismo entre os diferentes participantes, entre outros.

As classes relacionadas aos quadros de uma apresentação, como um *deck*, um quadro público ou privado, e os controles utilizados para visualizar objetos dessas classes estão contidos no pacote **Presentation.** Essas classes definem todo o modelo da apresentação colaborativa utilizada, e são exibidas na Figura 47.

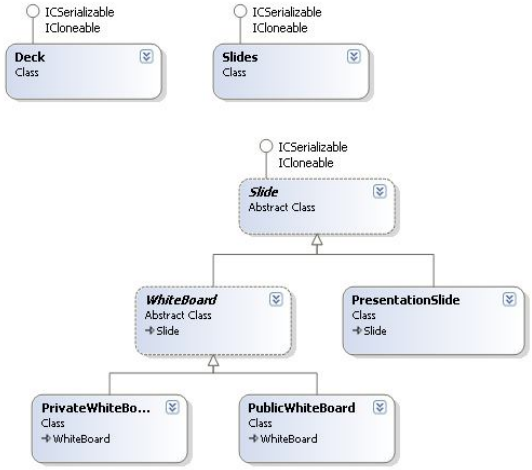

Figura 47 - As classes de uma apresentação colaborativa

O pacote **Session** contém classes relacionadas ao controle da sessão de colaboração, como a classe que representa a sessão de colaboração utilizada, e a classe relacionada a chave da sessão de colaboração.

O último pacote, chamado **User**, contém informações a respeito dos participantes e seus papéis em uma sessão de colaboração. As classes que representam seus papéis contêm métodos que definem quais mensagens cada tipo de participante pode enviar ou receber. Essas classes são exibidas na Figura 48.

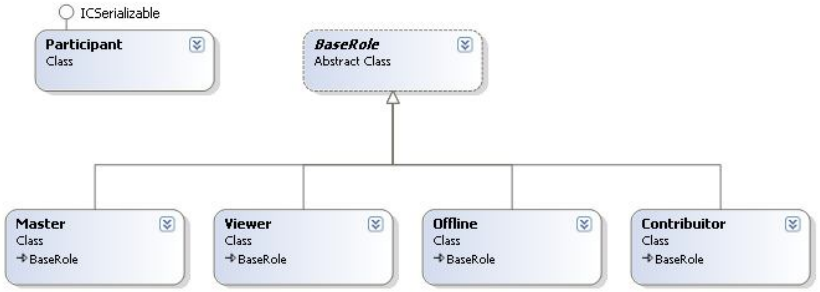

Figura 48 - Classes do pacote iPH.Commons.User

6.1.2.2. iPH.Desktop

Neste pacote estão contidas todas as classes específicas para a versão iPH – XP. A classe do formulário principal, chamado *MainForm*, estende o formulário *InteractivePresentationForm* e implementa seus métodos virtuais e abstratos. Basicamente, esta classe fornece apenas a interface com o usuário e o acesso ao arquivo de configuração da aplicação, e a partir das interações do usuário com esta interface, os métodos da classe pai são chamados. É também estendido o formulário de configuração da aplicação, pois para configurar as cores dos quadros públicos, são utilizados diálogos específicos para a plataforma utilizada.

6.1.2.3. iPH.Mobile

Este pacote é o correspondente do pacote iPH.Desktop para a versão iPH – Mobile. A classe *MobileMainForm* estende a classe *InteractivePresentationForm*, assim como o formulário de configuração, visto que os diálogos para a configuração de cor são específicos para a plataforma utilizada. Além destas duas classes, existem outras duas classes de formulários que são utilizadas para a formatação da fonte do texto a ser inserido em um quadro, e visualização das informações de um quadro de uma apresentação colaborativa.

7
Testes de Desempenho

Para avaliar a viabilidade do uso da aplicação iPH, foram realizados testes referentes ao tempo de transmissão de mensagens entre participantes com diferentes dispositivos, e testes referentes ao consumo de energia dos dispositivos durante o uso da aplicação.

7.1.1.
Envio e Recebimento de Mensagens

No iPH, várias mensagens são transmitidas entre os participantes durante uma sessão de colaboração. O tamanho de cada mensagem varia de acordo com seu tipo e seu conteúdo. Logo, é importante medir o tempo médio de transmissão destas mensagens em um cenário típico de uso da aplicação, como uma sala de aula, para assim verificar o desempenho da aplicação e do *middleware* de comunicação, o CompactConferenceXP.

Para a realização dos testes de desempenho da transmissão de mensagens foi montado um cenário com diferentes dispositivos em um ambiente com conexão *Ethernet* de 10 *Mbits* para computadores fixos e conexão wireless 802.11b de até 11 *Mbits* para os dispositivos portáteis. As medições de tempo foram realizadas para três operações básicas do iPH, que são:

- **Envio de Deck:** o dispositivo mestre envia *decks* de diferentes tamanhos (especificamente, 250 KB, 500 KB, 750 KB, 1 MB, 2 MB e 3 MB) para demais participantes;
- **Sincronização de Quadro:** o mestre muda para o próximo quadro, e para todos os participantes sincronizados com o mestre, mede-se o tempo até que todos os partcipantes também visualizem este próximo quadro;
- **Submissão de Contribuição:** um contribuidor submete uma contribuição ao mestre, e mede-se o tempo até que esta contribuição

é visualizada na aba de contribuições do mestre. Estas contribuições são padronizadas e possuem igual conteúdo (um retângulo feito com a *ink*, e um texto centralizado com seis caracteres).

Para medir o tempo de transmissão das mensagens, foi utilizada uma mensagem simples de confirmação de recebimento, que chamaremos de mensagem de confirmação, que foi criada especificamente para a realização dos testes, e que, portanto, não faz parte do protocolo original do iPH. Através dessa mensagem de confirmação, pode-se medir, no remetente, o intervalo de tempo entre o envio da mensagem original e o recebimento da confirmação. Portanto, assumindo que a latência da transmissão da mensagem de confirmação é pequena, praticamente desprezível, e igual para todos os dispositivos - uma vez que o tamanho da mensagem de confirmação é pequeno e igual para todos os testes - os resultados das medições ida-e-volta *(round-trip delay)* representam uma estimativa conservadora do tempo de transmissão da mensagem original. O propósito de calcular o tempo de recebimento de uma mensagem foi o de avaliar como a capacidade (diferenciada) dos diferentes dispositivos impacta no grau de sincronismo da colaboração, isto é, identificar quanto tempo demora até os dispositivos se sincronizarem. A Tabela 6 lista os dispositivos utilizados nos testes de transmissão.

Tabela 6 - Lista de dispositivos presentes no cenário de testes

Dispositivo	**Tipo**	**Papel**	**Memória RAM**	**Processador**	**Conexão**	**Qte.**
Toshiba Tablet PC Portegé M400-S4031	Tablet PC	Mestre	1 GB	Intel T2300 @ 1.66GHz	802.11b	1
HP iPAQ hx2400	Pocket PC	Contribuidor	64 MB	Intel PXA270 520mhz	802.11b	2
Qtek 9100	Smartphone	Contribuidor	64 MB	TI OMAP 850 200mhz	802.11b	1
Sony Vaio	Notebook	Contribuidor	256 MB	Intel Pentium 4 1.8GHz	802.11b	1
Computador Fixo (*Desktop*)	Desktop	Visualizador	1 GB	Intel Pentium 4 1.6 GHz	Ethernet	2

Em cada um dos testes foram realizadas 30 medições para cada um dos dispositivos, e o resultado é o tempo médio dessas medições. No caso dos *desktops* e dos *pocket pcs*, o resultado é a média dos resultados obtidos para cada um dos dispositivos. Como esperado, a capacidade computacional e a qualidade da conexão com a rede foram os principais fatores para a diferença dos resultados entre os dispositivos. Em todos os testes, computadores *desktops* obtiveram os melhores resultados, seguidos pelo *notebook*, *pocket pcs* e *smartphone*.

7.1.1.1. Envio de *Decks*

Como o iPH pode ser executado em dispositivos com diferentes capacidades de processamento e conexão com rede, este teste é importante pois o envio de um *deck* é a primeira operação realizada durante o uso da aplicação em uma sessão colaborativa. O envio de um *deck* consiste do envio de várias mensagens, cada qual contendo um quadro do *deck*. Para cada um dos quadros recebidos por um participante, este envia uma mensagem de confirmação de recebimento do determinado quadro. Portanto, para obter o tempo de envio de todo um deck a um participante, medimos, no mestre, a diferença entre o tempo de envio do primeiro quadro e o tempo do recebimento da confirmação do último quadro pelo participante.

A Figura 49 contém o gráfico com os resultados dos testes de envio de *decks*, para os diferentes dispositivos. Nota-se que aparentemente o tipo de conexão (Ethernet vs. *wi-fi* 802.11b) tem pouca influência sobre o desempenho do sistema (CCXP+iPH) uma vez que o *desktop* e o *notebook* apresentaram tempos muito próximos. Em compensação, percebe-se que o poder de processamento do dispositivo parece ter uma grande influência sobre o desempenho, haja vista as diferenças entre o *notebook*, o *pocket pc* e o *smartphone*.

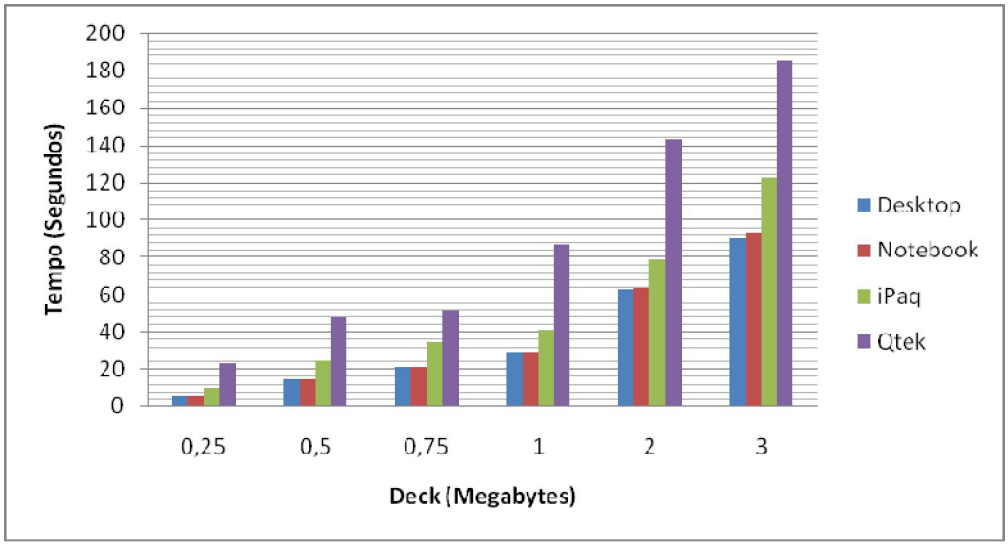

Figura 49 – Gráfico de envio de *decks* de diferentes tamanhos

Durante estes testes, foi percebido que algumas mensagens não foram recebidas por dispositivos conectados através da rede *wi-fi*, e somente foram computados os tempos quando **todas** as mensagens do *deck* eram recebidas pelos dispositivos. O número de mensagens perdidas cresceu de acordo com o tamanho do *deck* enviado, em uma relação direta com o número de mensagens enviadas, e o tipo de dispositivo. O *smartphone* teve uma média de 40% de mensagens perdidas no envio do *deck* de 3 MB, enquanto que os *pocket pcs* perderam somente 5% de mensagens, e o *notebook* perdeu 1% dessas mensagens. Contudo, esta perda de mensagens não compromete significativamente o uso do aplicativo iPH, pois na média, o tempo de envio total desse deck para o *smartphone* foi de três minutos. Logo, não é problema o usuário mestre enviar novamente o *deck* para os participantes, até que perceba que todos os dispositivos tenham recebido todos os quadros. Cabe observar ainda que no re-envio de um deck, apenas os quadros faltantes são incluídos, ou seja, não há a duplicação de quadros no dispositivo receptor.

7.1.1.2. Sincronização de um Quadro

O teste de sincronização de um quadro visa avaliar quanto tempo os diferentes dispositivos sincronizados com o mestre levam para se sincronizar

quando o mestre troca o quadro a ser visualizado (p.ex. avança ou retrocede um quadro no *deck*). Este tempo é a diferença entre o momento de envio da mensagem que solicita a visualização de um determinado quadro, e o momento do recebimento da confirmação relativo a essa mensagem.

O fato de esta mensagem ser bastante simples, contendo apenas o identificador do quadro a ser visualizado, foi determinante para que os tempos medidos fossem pequenos e muito semelhantes para todos os dispositivos, de fato, praticamente imperceptíveis durante utilização da aplicação. A Figura 50 exibe o gráfico contendo os tempos de sincronização para os diferentes dispositivos. Pode-se notar que a diferença entre o melhor e o pior tempo é desprezível.

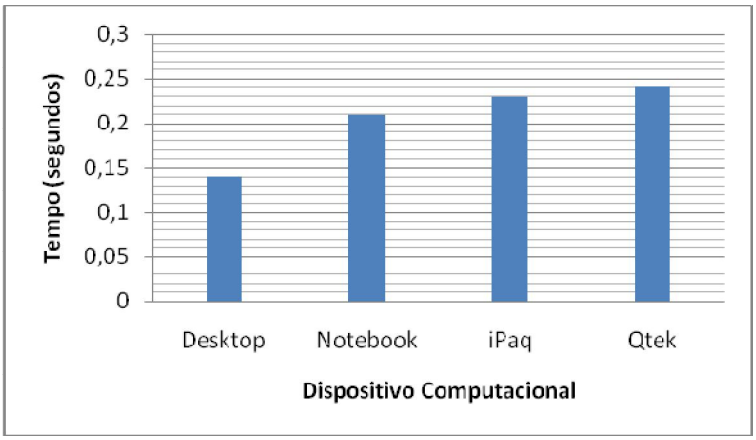

Figura 50 - Gráfico de cálculo de tempo de sincronização de quadros

7.1.1.3. Submissão de Contribuições

Enquanto que os outros dois testes estão relacionadas às operações realizadas pelo mestre, este teste (de submissão de contribuições) está focado na operação de um contribuidor, e visa medir o tempo que demora até o mestre receber uma contribuição, desde o momento de seu envio, pelo contribuinte. Para o teste, a contribuição enviada (para um *tablet pc* conectado na rede 802.11b) foi padronizada em dois objetos gráficos: um retângulo feito com a *ink*, e um texto centralizado. De fato, há pouca diferença no tamanho da contribuição enviada. A Figura 51 ilustra a contribuição em ambas as versões do iPH.

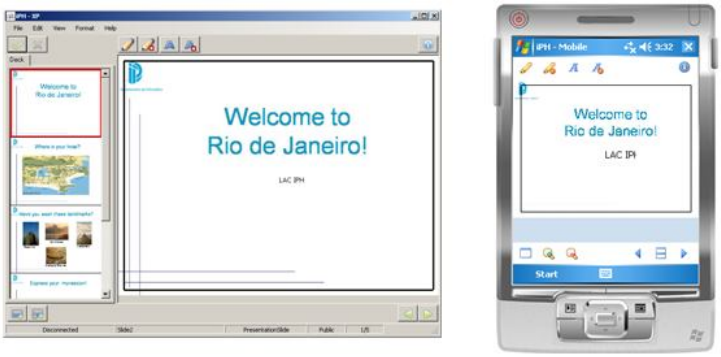

Figura 51 - Contribuição utilizada em teste de submissão

De acordo com o esperado, os dispositivos com maior capacidade foram os mais rápidos no envio das contribuições, e obtiveram resultados significativamente melhores do que os *handhelds*. Porém, estes últimos também tiveram tempos satisfatórios (3-4 segundos), que não prejudicam sua utilização em uma sessão colaborativa. A Figura 52 ilustra o gráfico com o tempo de recebimento das contribuições pelos diferentes dispositivos.

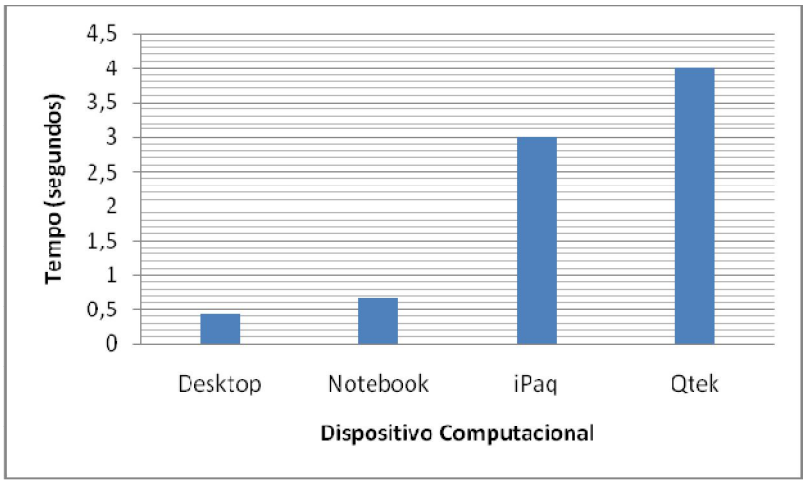

Figura 52 - Gráfico de cálculo de tempo de submissão de contribuição

7.1.1.4. Conclusões

Os testes de desempenho realizados visaram as operações mais importantes do iPH em uma sessão colaborativa: o envio de um *deck*, a sincronização de quadros e o envio de uma contribuição. Ficou atestado que o envio de um *deck* é definitivamente a operação mais demorada e com mais chances de falha. A forma

como ela é realizada – cada quadro é enviado em uma mensagem separada – apesar de causar a eventual perda de alguns quadros, principalmente em *decks* maiores, ainda é melhor do que o envio completo do deck em uma única mensagem. Esta alternativa, implementada em nossa versão inicial do iPH, mostrou uma latência tamanha (da ordem de dezenas de minutos) que tornaria o uso do aplicativo inviável em uma situação real. Além disso, quando há perdas de alguns quadros e o *deck* é re-enviado, somente as mensagens faltantes são recebidas e corretamente inseridas no *deck* em questão.

Ficou comprovado também que o iPH é igualmente adequado para dispositivos de menor poder computacional, tais como os *handhelds*, e que estes, apesar de apresentarem um desempenho inferior aos *notebooks* e *tablet pcs*, ainda assim são perfeitamente apropriados para uso do iPH em sala de aula ou reuniões. Além disso, percebeu-se que as diferenças entre as redes Ethernet e a rede *wi-fi* quase não impactam no desempenho do iPH, visto que aparentemente a comunicação através de *multicast* utilizada pelo CCXP funcionou igualmente bem em ambas as redes.

No entanto, deve-se observar que os testes realizados envolveram poucos dispositivos. Caso sejam utilizados mais do que uma dezena de dispositivos conectados na rede *wi-fi* 802.11b, é provável que se perceba uma queda de desempenho (e/ou uma maior taxa de falhas – perdas de mensagem) na comunicação pela rede sem fio, visto que os dispositivos móveis compartilharão a mesma conexão o que deverá gerar um número maior de colisões no protocolo de acesso ao meio (CSMA/CA), ocasionando retardos maiores para o envio de mensagens.

7.1.2. Nível de Energia

O teste para o cálculo do consumo de energia foi baseado no fato de que durante uma aula, os participantes geralmente utilizam seus dispositivos desconectados do cabo de força. Portanto, o propósito deste teste é avaliar durante quanto tempo os participantes podem utilizar o iPH continuadamente até que os dispositivos desliguem devido a falta de bateria. Neste teste foram utilizados um *tablet pc* e um *pocket pc*, pois além de serem dispositivos móveis, são próprios

para execução do iPH, visto que possuem suporte à *ink*. Durante o teste, os visores destes dispositivos permaneceram sempre acesos, como se estivessem sendo constantemente utilizados pelos participantes de uma aula.

Periodicamente, os dispositivos conectados a uma sessão de colaboração exercendo papel de mestre, executaram uma das seguintes ações:

1. Abrir um *deck* a partir de um arquivo;
2. Envio do *deck* completo para os demais participantes;
3. Envio mensagem informando conexão;
4. Envio de mensagem requisitando aos demais participantes que enviem mensagens de conexão;
5. Mudança para o próximo quadro do *deck*, exceto quando o quadro atual era o último quadro do *deck*. Neste caso, o primeiro quadro era visualizado.
6. Envio de contribuições aos demais participantes. Estes contribuições foram realizadas automaticamente de forma randômica, variando entre desenhos e textos.

7.1.2.1. Resultados

Foi possível constatar que o consumo da bateria do *tablet pc* foi consideravelmente maior do que o consumo da bateria do *pocket pc*. Após três horas de execução ininterrupta, a bateria do *tablet pc* se aproximou dos 10%, enquanto que a do *pocket pc* ainda estava em 60%. Estes valores, que talvez sejam influenciados pelas características dos dispositivos utilizados no teste, comprovam que o aplicativo iPH pode ser muito bem usado em uma aula de três horas ou mais. A Figura 53 mostra o gráfico criado a partir do resultado do teste realizado.

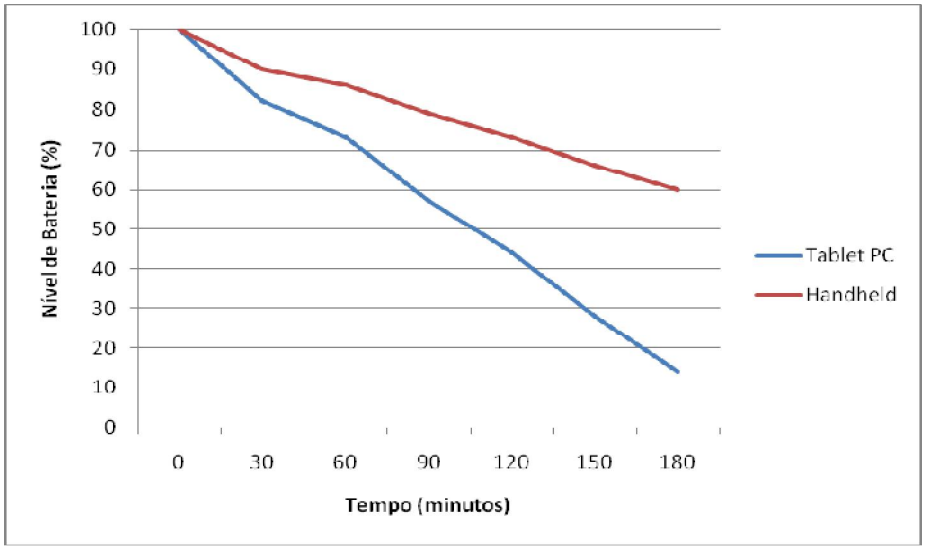

Figura 53 - Gráfico de consumo de bateria durante execução

Naturalmente, estes resultados podem variar significativamente dependendo da tecnologia utilizada na bateria (Ni-Cd, Ni-MH, Li-Ion), e do estado de conservação da bateria, dado que baterias mais utilizadas geralmente apresentam um poder de armazenamento menor de energia do que baterias novas.

8
Conclusões

Neste trabalho foi apresentado o **Interactive Presenter for Handhelds – iPH**. Esta aplicação oferece suporte ao compartilhamento e co-edição de apresentações, é sensível a informações de contexto computacional e está disponível em duas versões: uma para dispositivos como *notebooks* e *tablet pcs*, chamada iPH – XP; e outra versão para dispositivos mais limitados como *palmtops* e *smartphones*, chamada iPH - Mobile. O iPH constitui a principal contribuição e objetivo do trabalho, pois viabiliza a realização de experimentos de ensino interativo em salas de aula com diferentes tipos de dispositivos.

Para o desenvolvimento do iPH, houve a necessidade da criação de componentes responsáveis para determinadas funcionalidades da aplicação como a comunicação, a forma de edição (inserção e exclusão de contribuições), e acesso a informações de contexto. Essas necessidades podem ser encontradas em outras aplicações para colaboração móvel, e, portanto, os componentes correspondentes foram desenvolvidos para funcionarem de maneira independente, de modo a poderem ser reaproveitados. Os principais componentes criados foram:

- **CompactConferenceXP:** conjunto de APIs do *middleware* ConferenceXP adaptado para execução em dispositivos portáteis com poucos recursos (pouca memória, CPU mais lenta, etc.), e que executam apenas o .NET Compact Framework;
- **LAC.Contribs:** componente para inserção e exclusão de desenhos ou textos;
- **MoCA/WS:** *web service* que provê acesso aos serviços de informação de contexto oferecidos pela MoCA. A implementação deste *web service* foi necessária para permitir a interoperabilidade do código do iPH, na linguagem Visual C#, com o código Java dos serviços da MoCA.

Além do iPH, todos estes componentes também são considerados contribuições do trabalho apresentado, visto que quaisquer projetos em C# para

dispositivos móveis podem utilizá-los, seja para implementar formas de comunicação ou sincronização, capturar a entrada de dados gráfica dos usuários, ou para acessar informações de contexto dos dispositivos conectados a uma rede.

Durante a fase de desenvolvimento do iPH, várias questões complexas de usabilidade tiveram que ser discutidas, e as soluções acabaram sendo refletidas nas seguintes características da aplicação:

a) **Forma de visualização dos quadros de uma apresentação em dispositivos mais limitados:** foram criados controles visuais que são utilizados em ambas as versões do iPH. A similaridade na apresentação e funcionalidade desses controles garante uma mesma interpretação de um quadro independente do dispositivo em que está sendo utilizado;

b) **Formas de interação homem-máquina com as diferentes versões da aplicação:** as versões do iPH tem como objetivo oferecer o maior espaço possível ao usuário para visualização de um quadro. Além disso, foram definidas duas formas de edição: *ink*, para fazer desenhos, e texto, para ser inserido na coordenada escolhida;

c) **Modo como uma apresentação seria distribuída e sincronizada entre os participantes:** a apresentação é distribuída e sua visualização é controlada comandada pelo mestre, que pode inserir durante a apresentação quadros públicos e/ou privados. A sincronização é quebrada pelo participante (ao iniciar uma contribuição), e a re-sincronização pode ser solicitada pelo mestre, mas de fato precisa ser re-estabelecida explicitamente pelo participante contribuidor, garantindo assim que este não seja interrompido por uma mudança de quadro enquanto trabalha em sua contribuição;

d) **Visualização dos quadros da apresentação a partir de um projetor:** criou-se o papel do visualizador (*viewer*), dissociado do papel de mestre, mas cujo controle de sincronização é feito pelo mestre, para permitir que em determinados momentos da aula o mestre possa projetar um quadro para toda a classe, enquanto visualiza outro quadro (ou contribuição) em seu dispositivo;

e) **Uso de informações de contexto para adaptação:** foi definido que o mestre pode criar regras baseadas em condições de contexto computacional para habilitar e desabilitar determinadas funcionalidades dos participantes contribuidores.

Em relação ao desempenho do iPH, este apresentou resultados satisfatórios em suas diferentes versões. Diversos testes foram realizados para calcular o tempo da transmissão de mensagens por diferentes dispositivos. Esses testes foram realizados com dispositivos que incluíram desde computadores *desktops* até dispositivos como *palmtops* e *smartphones*. A principal diferença no desempenho desses dispositivos foi notada com relação ao tempo de transmissão da apresentação completa (todo o *deck* de quadros) a ser compartilhada. Quanto maior é o *deck*, mais tempo os dispositivos do tipo *handheld* demoram a recebê-la. Este é o único momento em que se pode notar um pior desempenho destes dispositivos em relação aos *tablet pcs* e *notebooks*. A partir do momento em que a apresentação tiver sido difundida para todos os dispositivos, o envio de contribuições e a sincronização entre quadros acontecem de maneira quase instantânea.

É interessante relatar que na nossa primeira versão do iPH, antes da realização dos testes, uma apresentação era enviada em uma única mensagem. Esta implementação se mostrou inviável visto que o CompactConferenceXP implementa protocolo RTP (Real-Time Transport Protocol) [RTP, 2007], que é adequada para a transmissão de *streams* de vídeo em termo real, e utiliza o algoritmo Reed-Solomon para correção de erros na transmissão de quadros (de vídeo) [Wicker, 1999]. Este algoritmo é polinomial, e para dispositivos mais limitados como *handhelds*, a execução do mesmo resultou em uma sensível queda de desempenho, tornando praticamente inviável a utilização destes dispositivos, haja visto que um deck de 1 MB demorava aproximadamente 15 minutos para ser recebido por um *handheld*. Para contornar este problema, na versão atual do iPH re-implementamos a transmissão de decks com o envio de um quadro por mensagem, e o que gerou resultados bem aceitáveis para todos os dispositivos, como relatado no capítulo 7.

Através do desenvolvimento e testes do iPH, julgamos que todos os objetivos previstos inicialmente para este trabalho foram cumpridos de forma satisfatória, a saber: o suporte a apresentações, o compartilhamento de

apresentações, a sensibilidade a informações de contexto e a execução em diferentes dispositivos.

Por fim, deve-se observar que devido à utilização do CompactConferenceXP para comunicação entre os participantes, o roteador da rede *wi-fi* utilizada pelos dispositivos móveis que executam o iPH deve necessariamente ter suporte a endereços *multicast*.

8.1. Trabalhos Futuros

Este trabalho apresentou uma aplicação e componentes de *middleware* que estão prontos para serem evoluídos em trabalhos futuros. Estes podem ser agrupados em quatro categorias: componentes, aplicação, arquitetura e experiências com o uso do iPH em sala de aula.

8.1.1. Componentes de Middleware

Em nível de componentes, trabalhos podem ser desenvolvidos para evoluir as *APIs* do CompactConferenceXP, de acordo com a evolução do *middleware* original ConferenceXP, como também para melhorar o desempenho em dispositivos do tipo *handheld*. Existem serviços oferecidos por este *middleware* que ainda não foram adaptados e testados para a utilização em dispositivos móveis.

Sobre a LAC.Contribs, a qualidade dos desenhos feitos é um pouco inferior (curvas menos suaves) à qualidade dos desenhos capturados com a Microsof.Ink, que está disponível somente para o .NET Framework. Seria interessante melhorar esta qualidade, assim como criar uma opção de borracha que apague somente a parte do desenho (os segmentos da curva) que esteja em contato com a borracha. Atualmente, a borracha apaga a linha inteira desenhada que ela “toca”.

O MoCA/WS oferece chamadas e subscrições apenas aos serviços CIS e LIS da MoCA, que continuam em constante evolução. Portanto, este *web service* deveria ser estendido para contemplar os demais serviços da MoCA, acompanhando a evolução desta arquitetura.

8.1.2. Aplicação

Em nível de aplicação, vários elementos podem ser melhorados principalmente com relação à interação com o usuário. Por exemplo, o painel que exibe o conjunto de quadros de uma apresentação não exibe, para cada quadro, as contribuições já realizadas, como ocorre nas outras ferramentas apresentadas no capítulo 2. Isto é importante, pois facilita a identificação do quadro que o usuário deseja visualizar no painel de visualização de quadro.

Várias novas formas de colaboração podem ser criadas, como por exemplo, a inserção de um *chat*, permitindo uma comunicação entre participantes que não estejam presentes no mesmo local. Outra forma de colaboração poderia ser o envio de enquetes (questionários eletrônicos) e questões de múltiplas escolhas pelo mestre para os demais participantes. Ao receber as respostas, o mestre poderia avaliar instantaneamente o grau de compreensão do assunto pelos estudantes a partir de gráficos.

A funcionalidade de persistência de uma apresentação colaborativa é importante para a utilização do iPH. Com ela, os participantes poderiam salvar a apresentação em um arquivo para visualizar posteriormente, além de poder enviar este arquivo para outras pessoas que estejam interessadas em no conteúdo ministrado. Para facilitar o aprendizado dos participantes, um módulo para rever passo-a-passo o andamento de uma apresentação compartilhada, como o Presenter Playback (vide seção 2.1), poderia ser desenvolvido. Por exemplo, um participante poderia necessitar visualizar uma apresentação em que esteve ausente. Também é possível aperfeiçoar a navegabilidade da ferramenta ao oferecer ao usuário várias visões independentes dos quadros da apresentação. Assim, um usuário possuiria uma visão que estaria sempre sincronizada com o mestre, e uma ou mais visões para edições de seus quadros locais.

Outra extensão seria a criação de um diretório de sessões de colaboração (ativas e inativas) com seus documentos associados (e.g. *deck* principal + contribuições), e que pudesse ser consultado pelos usuários para decidirem qual sessão desejam participar. Atualmente, o conceito de sessão do iPH está fortemente associada ao endereço *multicast* utilizado para a troca de mensagens entre o grupo de participantes. Assim, um usuário que não esteja presente no

mesmo local que o mestre da apresentação, pode desconhecer o endereço *multicast* utilizado, e ficar impossibilitado de participar da sessão.

O uso de informações de contexto pode ser mais bem explorado no futuro. Atualmente, apenas a funcionalidade de submissão de contribuições pode ser associada a uma regra de contexto, e novas formas de adaptação poderiam ser criadas para melhorar a interação com o usuário. O modelo utilizado de forma experimental nesta versão do iPH, onde o mestre cria as regras de contexto durante a sessão de colaboração, pode ser evoluído para algo de mais fácil gerenciamento. Por exemplo, as regras de contexto poderiam fazer parte de uma pré-configuração da sala de aula, e provavelmente seriam formuladas por um administrador. Ao mestre caberia meramente o papel de desabilitar e habilitar estas regras.

8.1.3. Arquitetura

Atualmente o iPH possui a limitação de poder ser utilizado somente por dispositivos que estejam em uma mesma rede, e que suporte *multicast*. Como visto no Capítulo 2, esta é a mesma limitação do Classroom Presenter, que originou o Ubiquitous Presenter (UP). O UP utiliza *web services* para fazer com que usuários participem de uma sessão colaborativa a partir de seus *browsers*, sem a necessidade de instalação de aplicativos e conexão à mesma rede dos demais participantes. Logo, a arquitetura do iPH pode evoluir para outros modos de comunicação além do fornecido pelo CompactConferenceXP.

Outro trabalho interessante seria criar uma arquitetura independente para o desenvolvimento de aplicações colaborativas que forneça toda a estrutura de comunicação, como um sistema de componentes colaborativos. A arquitetura do sistema ficaria responsável pela comunicação e entrega de mensagens aos devidos componentes. O compartilhamento e co-edição de apresentações seria somente um dos componentes criados para este sistema. Vários outros componentes poderiam ser desenvolvidos e utilizados pela arquitetura, como:

- Gerenciamento de grupos de usuários;
- Troca de mensagens instantâneas;
- Transferência de arquivos;

- Compartilhamento e edição de textos;

Os componentes desenvolvidos deveriam implementar interfaces estabelecidas pela arquitetura e dependências entre esses componentes poderiam existir. Por exemplo, o componente de troca de mensagens só poderia ser utilizado em conjunto com o componente de gerenciamento de usuários. A Figura 54 exibe a idéia inicial deste sistema, separando os componentes que são carregados e as funcionalidades básicas de comunicação fornecidas pela arquitetura.

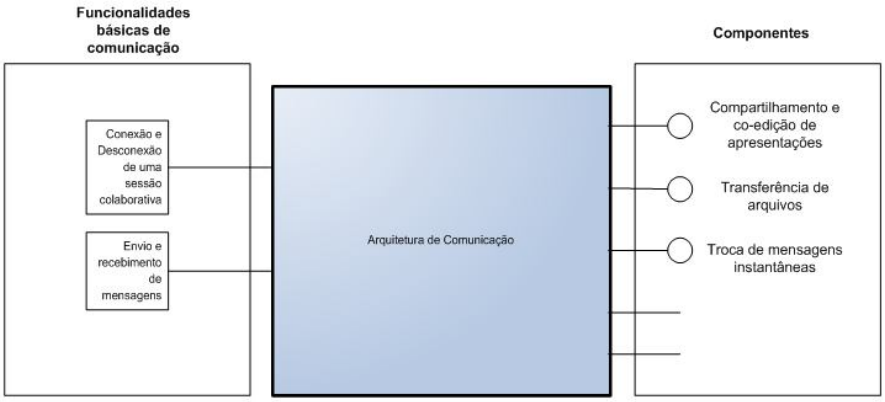

Figura 54 - Um sistema de componentes colaborativos

8.1.4. Experiências com o uso do iPH

Em nível de experiência prática de uso em uma sala de aula, precisa-se ainda fazer testes que comprovem que a utilização do iPH efetivamente motiva uma maior participação dos estudantes, aumentando o grau de compreensão dos assuntos ministrados. Tais experimentos de uso do iPH em aulas poderia resultar em uma série de sugestões acerca de possíveis melhorias da interação com o usuário, e testaria a execução do iPH e transmissão de mensagens do CompactConferenceXP em ambientes com um grande número de dispositivos móveis presentes.

9
Referências Bibliográficas

[Anderson, 2004a] Anderson, R., Anderson, R., Simon, B., Wolfman, S. A., VanDeGrift, T., and Yasuhara, Ken. Experiences with a Tablet PC Based Lecture Presentation System in Computer Science Courses. 35th SIGCSE, Março 2004.

[Anderson, 2004b] Anderson, R. J., Hoyer, C., Wolfman, S., and Anderson, R. A Study of Digital Ink in Lecture Presentation. In Proceedings of CHI 2004 (Vienna, Austria), Abril 24-29, 2004.

[Anderson, 2006] Richard Anderson, Ruth Anderson, O. Chung, K. M. Davis, P. Davis, C. Prince, V. Razmov and B. Simon**. "**Classroom Presenter - A Classroom Interaction System for Active and Collaborative Learning" *WIPTE 2006*, 2006.

[Anderson, 2007] Anderson, R., et al. Supporting active learning and example based instruction with classroom technology. Technical Symposium on Computer Science Education Proceedinds of the 38th SIGCSE technical symposium on Computer science education 2007.

[Beavers, 2004] Beavers, J., Chou, T., Hinrichs, R., Moffatt, C., Pahud, M., Powers, L., and Van Eaton, J., The Learning Experience Project: Enabling Collaborative Learning with ConferenceXP, Microsoft Research Technical Report MSRTR- 2004-42, Abril 2004.

[Berque, 2004] Berque D., Bonebright T., and Whitesell M. Using Pen-based Computers Across the Computer Science Curriculum. 35th SIGCSE, 2004.

[Coutaz, 2005] Coutaz, J., Crowley, J., Dobson, S. and Garlan D. Context is key. Communications of the ACM, 48(3):49-53, 2005.

[C#, 2007] *Visual C# Developer Center*, http://msdn.microsoft.com/vcsharp/. Acesso em Maio 2007.

[CFNET, 2007a] .*NET Compact Framework*, http://msdn.microsoft.com/netframework/programming/netcf/ . Acesso em Maio 2007.

[CFNET, 2007b] .*NET Compact Framework Architecture.* http://msdn2.microsoft.com/en-us/library/9s7k7ce5(vs.80).aspx. Acesso em Junho 2007.

[CFNET, 2007c] *Differences with the .NET Framework, 2007,* http://msdn2.microsoft.com/en-us/library/2weec7k5(vs.80).aspx. Acesso em Maio 2007.

[ClassroomPresenter, 2007] *Classroom Presenter*, University of Washington 2006, http://www.cs.washington.edu/education/dl/presenter/. Acesso em Junho 2007.

[CompactFormatter, 2007] *CompactFormatter: A generic formatter for the .NET Compact Framework.* http://www.freewebs.com/compactFormatter/. Acesso em Maio 2007.

[ConferenceXP, 2006] *Conference XP*, Microsoft Research, Redmond, WA, USA, 2007, http://www.conferencexp.net/. Acesso em Junho 2007.

[Dey, 2000] Dey, A. and Abowd, G., Towards a Better Understanding of Context and Context-Awareness, Workshop on the what, who, where, when and how of context-awareness at CHI 2000, Abril 2000.

[DyKnow, 2007] *DyKnow.* http://www.dyknow.com/. Acesso em Junho 2007.

[Fuchs, 2001] Fuchs, L., Poltrock, S., and Wetzel, I. TeamSpace: An Environment for Team Articulation Work and Virtual Meetings, to appear: Proc. First

International IEEE Workshop on Web-based Collaboration, Munich, Germany, Setembro 2001.

[Gamma, 1995] Gamma, E., Helm, R., Johnson, R., and Vlissides, J..Design Patterns: Elements of Reusable Object-Oriented software. Addison-Wesley, 1995.

[Geyer, 2001] W. Geyer et al. A Team Collaboration Space Supporting Capture and Access of Virtual Meetings. In Proceedings of the 2001 International ACM SIGGROUP Conference on Supporting Group Work, 2001.

[Golub, 2004] Golub, E. Handwritten Slides on a TabletPC in a Discrete Mathematics Course. 35th SIGCSE, 2004.

[IETF, 2007] Internet Engineering Task Force. http://www.ietf.org/. Acesso em Junho 2007.

[Iles, 2002] Iles, A., Glaser, D., Kam, M. & Canny, J. Learning via distributed dialogue: Livenotes and handheld wireless technology. Proc. CSCL ’02, pp. 408-417.

[Java, 2007] *Java Technology*. http://java.sun.com/. Acesso em Junho 2007.

[Kadous, 2004] Mohammed Waleed Kadous, Claude Sammut, "MICA: Pervasive Middleware for Learning, Sharing and Talking," *percomw*, p. 176, *Second IEEE Annual Conference on Pervasive Computing and Communications Workshops*, 2004.

[Kam, 2002] M. Kam, O. Tarshish, D. Glaser, A. Iles, and J. Canny, Communicating through handheld wireless tablets: livenotes and shared group awareness, in Supplemental Proceedings of ACM Conference on Computer Supported Cooperative Work, 2002, pp. 143-145.

[Kam, 2005] Kam, M., Wang, J., Iles, A., Tse, E., Chiu, J., & Glaser, D. et al. (2005). Livenotes: a system for cooperative and augmented note-taking in

lectures. CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems, Portland, Oregon, USA, 531-540.

[Livenotes, 2007] *Livenotes*, University of California, Berkeley. http://www.cs.berkeley.edu/~mattkam/livenotes/. Acesso em Junho 2007.

[Maite, 2003] W. M. Waite, M. H. Jackson, and A. Diwan. The Conversational Classroom. In Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education, Reno, Nevada, pages 127{131. ACM Press, New York, Feb. 2003.

[Malcher, 2006] M. Malcher, MoCA/WS: Integração do middleware MoCA com Web Services, *Documento da Disciplina Projeto Final de Programação – Mestrado PUC-Rio*, Rio de Janeiro, Julho 2006.

[Moca, 2007] Mobile Collaboration Architecture, http://www.lac.inf.puc-rio.br/moca/. Acesso em Julho 2007.

[Multicast, 2007] *Using ConferenceXP in a Multicast Network*, http://research.microsoft.com/conferencexp/setup_multicastnetwork.aspx. Acesso em Junho 2007.

[Nascimento, 2006] F. Nascimento, V. Sacramento, G. Baptista, H. Rubinsztejn, M. Endler, Desenvolvimento e Avaliação de um Serviço de Posicionamento Baseado em IEEE 802.11, *Proc. of the XXIV Brazilian Symposium on Computer Networks (SBRC 2006)*, (short paper), Curitiba, May 2006.

[NET, 2007] *.NET Framework*. http://msdn.microsoft.com/netframework/. Acesso em Junho 2007.

[Nilson,2005] Nilson L., Weaver, B. (eds), Enhancing Learning with Laptops in the Classroom. New Directions for Teaching and Learning, Jossey-Bass, San Francisco, 2005.

[OpenNETCF, 2007] OpenNETCF.org. http://www.opennetcf.org/. Acesso em Maio 2007.

[Prekop, 2003] Prekop, P. and Burnett, M. Activities, context and ubiquitous computing, Special Issue on Ubiquitous Computing Computer Communications, Vol. 26, No. 11, pp.1168–1176.

[Richter, 2001] Richter, H., Abowd, G., Geyer, W., Fuchs, L., Daijavad, S., Poltrock, S., Integrating Meeting Capture within a Collaborative Environment, to appear: Proc. Ubicomp 2001, ACM Conference on Ubiquitous Computing, Atlanta, GA, USA, Setembro 30 - Outubro 2, 2001.

[Rößling, 2004] Rößling, G., Trompler, C., Muehlhäuser, M., Köler, S., Wolf, S. Enhancing Classroom Lectures with Digital Sliding Blackboards. 9th ITICSE, Junho 2004.

[RTP, 2007] RFC 3550 - RTP: A Transport Protocol for Real-Time Applications. http://www.faqs.org/rfcs/rfc3550.html. Acesso em Junho 2007.

[Sacramento, 2004] V. Sacramento, M. Endler, H.K. Rubinsztejn, L.S. Lima, K. Gonçalves, F.N.do Nascimento, G. Bueno, MoCA: A Middleware for Developing Collaborative Applications for Mobile Users *IEEE Distributed Systems Online, ISSN 1541-4922, vol. 5, no. 10, October, 2004*

[Salber, 2001] Salber, D., Siewiorek, D. P., and Smailagic, A. Supporting Mobile Workgroups on a Wireless Campus. In Proc. Int. Workshop on Human Computer Interaction with Mobile Devices.

[Smailagic, 2001] A. Smailagic, D. P. Siewiorek, J. Anhalt, F. Gemperle, D. Salber, S. Weber, J. Beck, and J. Jennings, "Towards Context Aware Computing: Experiences and Lessons," *IEEE Journal on Intelligent Systems*, vol. 16, pp. 38-46, 2001.

[SOAP, 2007] Simple Object Access Protocol (SOAP) 1.1. http://www.w3.org/TR/2000/NOTE-SOAP-20000508/. Acesso em Junho 2007.

[TabletMylarSlides, 2007] *The Tablet Mylar Slides Classroom Presentation System,* University of Maryland. http://www.cs.umd.edu/~egolub/TabletMylarSlides/. Acesso em Junho 2007.

[UbiquitousPresenter, 2007] *Welcome to UCSD Ubiquitous Presenter*, University of California, San Diego. http://up.ucsd.edu/. Acesso em Junho 2007.

[XML, 2007] *Extensible Markup Language (XML).* http://www.w3.org/XML/. Acesso em Junho 2007.

[Wicker, 1999] Wicker, S. B., Bhargava, V. K. Reed-Solomon Codes and Their Applications, John Wiley & Sons, Inc., New York, NY, 1999.

[Wilkerson, 2005] M. Wilkerson, W. Griswold, and B. Simon. Ubiquitous Presenter: Increasing Student Access and Control in a Digital Lecturing Environment. In SIGCSE'05, pp.116-120.

10
Anexos

10.1.
O aplicativo iDeck

O iDeck é onde são criados os arquivos de *deck* (*Interactive Deck Document*, extensão .idd) utilizados pelo iPH. Este aplicativo é utilizado pelo usuário que fará o papel de mestre para criar a apresentação que será compartilhada por todos os participantes em uma sala de aula. A Figura 55 exibe a tela principal do iDeck.

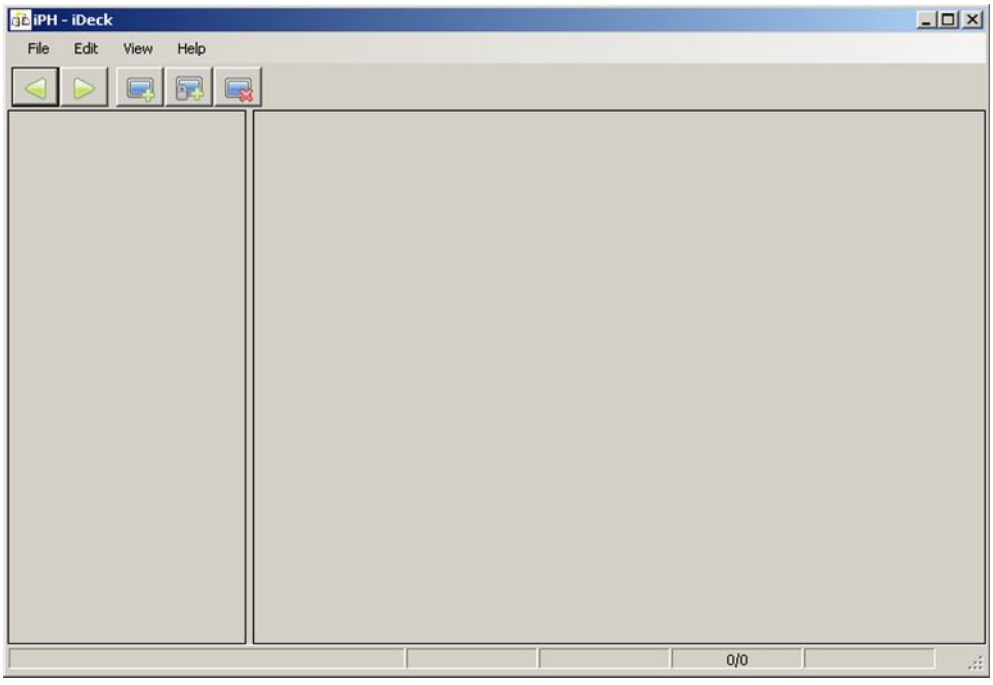

Figura 55 - O aplicativo iDeck

Com o iDeck, o usuário pode adicionar e remover e quadros públicos e privados (como exibido na Figura 56), configurar estes quadros (cor, título e observações) e inserir quadros a partir de uma apresentação (PowerPoint® ou

outros tipos), além de opções de salvar o *deck* criado, e editar *decks* existentes. Opções de navegação e de copiar e colar quadros também estão disponíveis.

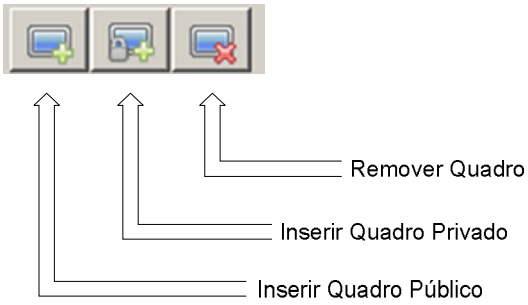

Figura 56 - Opções de inserir e remover quadros do iDeck

Para inserir os quadros de uma apresentação, basta que o usuário escolha o arquivo da apresentação, e todos os quadros da mesma serão inseridos. É importante ressaltar que todas as animações da apresentação são descartadas, pois o iDeck considera cada quadro como uma imagem *Jpeg*. A Figura 57 exibe a opção de *menu* para inserir uma apresentação.

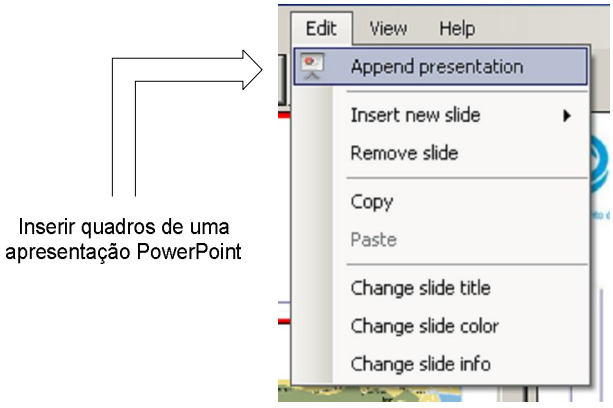

Figura 57 - Inserir uma apresentação no iDeck

A Figura 58 exibe uma apresentação visualizada no iDeck. O usuário pode visualizar as informações do quadro atual, assim como seu título, se é público ou privado, e qual sua ordem no conjunto de quadros. Este *deck* pode ser salvo e utilizado pelo participante mestre no iPH.

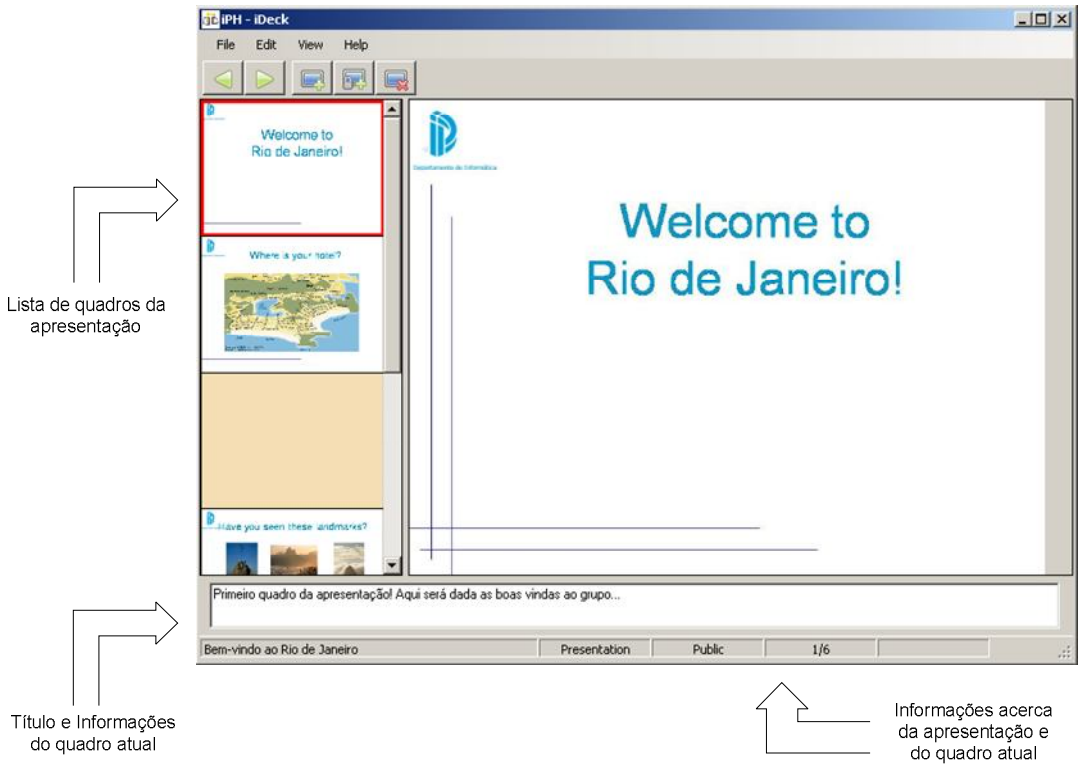

Figura 58 - Uma apresentação visualizada no iDeck