4
A Infra-estrutura de Execução ActivePresentation

Vimos que o controle coordenado, seja interativo (Seções 3.1, 3.3, 3.4) ou automático (Seção 3.2), de aplicações multimídia em um espaço ativo abre novas possibilidades para o uso desses espaços, tal como a orquestração de uma apresentação multimídia distribuída. Além disso, através da interligação de diferentes salas *multifuncionais*, essa mesma infra-estrutura permite a coordenação de aplicações em localidades distintas. A capacidade de interação do usuário sobre o andamento de uma apresentação ou aplicação, não importando sua localidade, poderia também permitir a cooperação de grupos para a realização de trabalho cooperativo síncrono à distância.

Portanto, a utilização plena desses ambientes deve contemplar a interação do usuário, que através dos múltiplos dispositivos de entrada pode conduzir dinamicamente uma determinada atividade. Para lidar com as múltiplas interfaces de entrada e saída dos vários dispositivos presentes em tais salas sem sobrecarregar o usuário, um sistema computacional precisa ter um certo grau de autonomia, seja através de um sistema de captura de intenção do usuário [27] ou através de uma programação da utilização do ambiente [26, 44].

A construção de tais sistemas autônomos depende, entre outras funcionalidades, da comunicação entre dispositivos computacionais, do intercâmbio de mensagens entre aplicações e da publicação de informações, tais como arquivos de dados, a disponibilidade de recursos computacionais (software e hardware) e o estado de execução de aplicativos. Tais funcionalidades são tipicamente oferecidas por middlewares para espaços ativos, tais como Gaia [26], Aura [27] e BEACH [44].

A infra-estrutura *ActivePresentation*, apresentada neste trabalho, não se propõe a ser um middleware para espaços ativos, mas sim a complementar tais sistemas, organizando e estruturando aplicações para a realização de apresentações multimídia distribuídas. Para isso, nos baseamos no projeto Gaia, citado anteriormente.

Outra decisão de projeto foi que, em vez de desenvolvermos aplicações totalmente novas, optamos por reutilizar ao máximo componentes e aplicações já existentes. Essa abordagem, além de reduzir drasticamente o tempo de desenvolvimento dos protótipos para a experimentação das idéias investigadas, permite que os usuários utilizem aplicações *single-user* [32] com as quais já estão familiarizados. Assim, boa parte da infra-estrutura *ActivePresentation* é destinada a oferecer mecanismos de integração e controle de aplicações e componentes de software já existentes.

Essa integração de diferentes aplicações e componentes de software para compor uma nova aplicação para salas *multifuncionais* pode ser uma tarefa bem complexa. Percebe-se que cada aplicação tipicamente oferece sua própria interface de controle programático, além de adotar diferentes tecnologias de middleware, tais como Java, COM, .NET e CORBA. Uma abordagem comum para essa integração é definir interfaces padronizadas entre as diferentes aplicações e oferecer adaptadores que façam as conversões necessárias entre as interfaces. Neste trabalho fazemos uso de adaptadores para tornar possível interligar o documento multimídia à interface remota da aplicação.

Para tratar do problema de interoperabilidade entre as diferentes tecnologias de middleware, *ActivePresentation* utiliza como mecanismo básico de composição a ferramenta LuaOrb [13, 9]. LuaOrb é uma ferramenta para o desenvolvimento de aplicações baseadas em componentes de software que oferece um modelo dinâmico para a utilização e implementação de componentes baseados em padrões da indústria, tais como CORBA, Java, COM e .NET. Além disso, LuaOrb oferece, através da criação de pontes dinâmicas entre componentes de diferentes tecnologias, um mecanismo simples e flexível de interoperabilidade.

4.1 Arquitetura Geral

O modelo apresentado na Figura 4.1 mostra as entidades envolvidas em uma apresentação: alguns *Nós de Apresentação e Controle*, um *Gerente de Grupo* e um *Nó Coordenador*. Os *Nós de Apresentação e Controle* possuem um serviço de *Controle de Recursos* e uma *Aplicação*; o *Gerente de Grupo* possui *Grupos de Aplicações* e um *Observador de Grupo*; o *Nó Coordenador* possui um *Observador* e uma aplicação que serve de *Coordenador*.

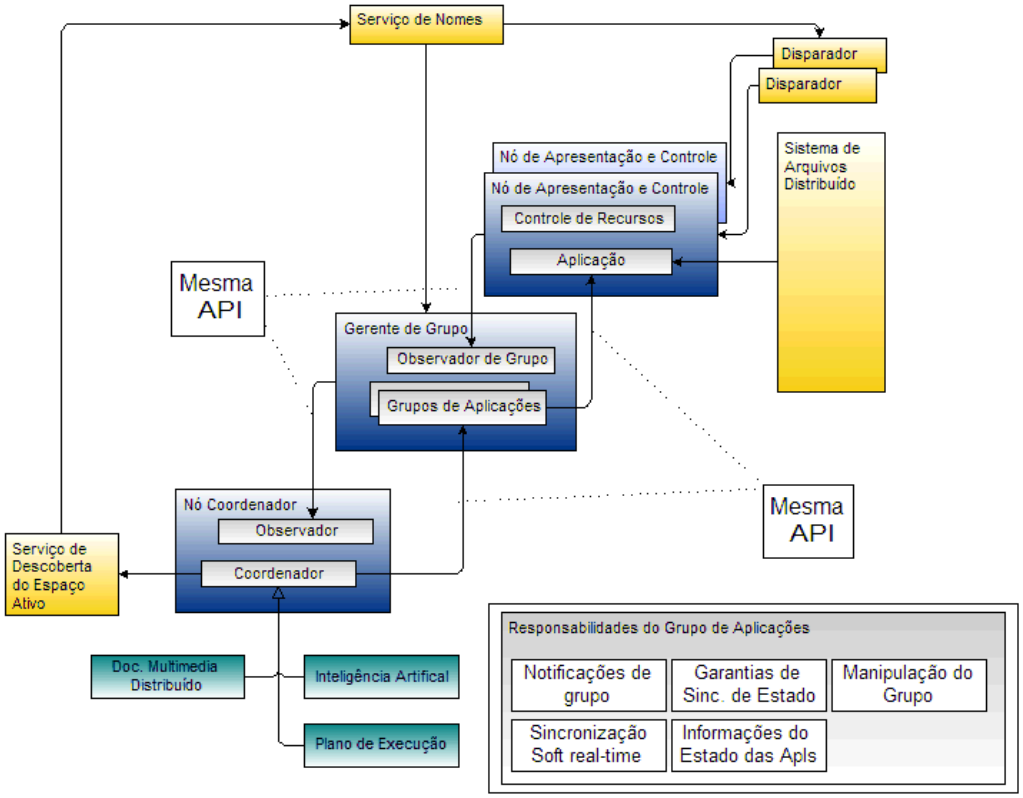

Figura 4.1: Arquitetura Geral do Sistema

Além dessas entidades, os seguintes serviços (baseados no sistema de computação ubíqua Gaia [26]) oferecem suporte à execução: *Sistema de Arquivos Distribuído*, *Serviço de Adaptação da Sala Multifuncional*, *Serviço de Nomes* e um serviço *Disparador* de aplicações.

4.1.1 Nó de Apresentação e Controle

O *Nó de Apresentação e Controle* tem como responsabilidade facilitar a distribuição de uma aplicação, permitindo que sua execução seja conduzida e observada remotamente por outras aplicações. Desta forma, o *Nó de Apresentação e Controle* oferece funcionalidades da infra-estrutura para a aplicação, tais como conexão com o *Gerente de Grupo*, controle dos observadores interessados em notificações sobre mudanças em seu estado e obtenção e disponibilização de arquivos de dados.

Além disso, o *Nó de Apresentação e Controle*, através de sua conexão com o *Gerente de Grupo*, provê acesso ao dispositivo, oferecendo a possibilidade de terminar a aplicação e seu processo. Outra função do *Nó de Apresentação e Controle* é receber os estados de execução que o documento mul-

timídia quer observar. Esses estados são verificados através desse invólucro que por sua vez repassa as informações ao interpretador do documento.

A inclusão de uma aplicação em um *Nó de Apresentação* é simples: ele exige a construção de uma fina camada de comunicação entre a aplicação e o nó, a exportação da interface de controle da aplicação no *Gerente de Grupo* (e.g. transformar a IDL COM oferecida pelo Internet Explorer® em uma IDL CORBA) e a implementação de um adaptador para o futuro formatador do documento multimídia. Como exemplo apresentamos no Segundo Apêndice a IDL COM do Internet Explorer® e a transformação dessa IDL para CORBA.

O conceito de *Nó de Apresentação e Controle* também permite que sobre a aplicação seja construído um Serviço de Presença através do mecanismo de *heartbeat*, como descrito no Capítulo 2.

Controle de Recursos

Através do módulo de *Controle de Recursos* é possível que uma aplicação reserve tempo de CPU individualmente para suas necessidades. No atual estágio de desenvolvimento, estamos interessados apenas em alocar os recursos individuais de um dispositivo computacional de forma a garantir a fatia de tempo necessária para a execução de nossas apresentações multimídia.

Atualmente, o maior fator que causa problemas de sincronização entre as aplicações (como analisado por Diwakar Gupta [31] e comprovado em nossos experimentos) é exatamente a falta de tempo de processamento. Em nossos protótipos experimentamos tanto a utilização do sistema *Dynamic Soft Real Time CPU Scheduler* (DSRT) [14] quanto a utilização direta do nível de prioridade de execução do Windows, ambos oferecendo resultados similares para nossos propósitos. A alocação de recursos se mostrou bastante útil principalmente durante a reprodução de uma mídia contínua sincronizada entre diferentes computadores e sendo executada em uma CPU sobrecarregada com outros processos.

Aplicação

Uma *Aplicação* é um software (seja de código aberto ou não) que oferece controle programático de sua execução. Esse controle deve ser feito através de uma interface de automação que, além de oferecer métodos para a realização de tarefas, recebe eventos de ações do usuário sobre a

sua interface gráfica. Além disso, a aplicação deve permitir ser iniciada e finalizada programaticamente.

Através da distribuição da interface de automação (se já não for distribuída), a aplicação poderá ser controlada remotamente; nota-se que essas chamadas remotas geralmente irão exigir baixa comunicação, oferecendo uma solução interessante para ambientes com limitada capacidade de rede.

Em nossos protótipos utilizamos aplicações Microsoft como o Windows Media Player® e o PowerPoint®. A distribuição da interface de automação dessas aplicações, oferecida através de COM, foi feita através da conversão de sua interface para CORBA. Observamos a possibilidade de automação desse processo, possibilitando que futuramente a inclusão de aplicações seja feita de forma automática através da conversão das IDLs de COM para CORBA.

4.1.2 Gerente de Grupo

O *Gerente de Grupo* centraliza as informações a respeito das aplicações em execução nos nós. São armazenados um identificador único do nó e uma referência remota para a aplicação que está em execução. Através dessas informações, o *Gerente de Grupo* pode responder perguntas como quais nós estão em operação, quais aplicações estão sendo executadas, quais grupos estão ativos, entre outras. Além disso, o *Gerente de Grupo* notifica observadores sobre registro e desregistro dos nós.

Finalmente, o *Gerente de Grupo* oferece controle de réplicas de aplicações abstraindo a multiplicidade. Isto é feito através de uma classe que obedece ao padrão Proxy [5] onde chamadas são recebidas e repassadas a cada membro do grupo. Desta forma, comandos efetuados sobre uma referência de grupo são repetidos em cada aplicação. De forma similar, eventos gerados por uma aplicação são repassados para o grupo que os analisa e os repassa como um evento de grupo. Apresentamos mais detalhes na próxima Seção.

Grupos de Aplicações

Para permitir o controle transparente e simultâneo de múltiplas aplicações, a infra-estrutura *ActivePresentation* provê um mecanismo de controle de réplicas através de grupos. Um grupo controla cada aplicação através da mesma interface oferecida por ela. Comandos enviados para um

grupo são repassados individualmente para cada aplicação com os mesmos parâmetros. Nota-se que essa mesma chamada pode também ser feita para uma aplicação individualmente.

Para otimizar a execução das chamadas sobre um grupo, evitamos fazer chamadas síncronas dos comandos enviados para os membros, o que previne o acúmulo de atrasos. Em vez disso, chamamos assincronamente o método para cada membro e posteriormente recuperamos a resposta de cada um deles. Desta forma, garantimos um menor retardo entre as chamadas devido à paralelização da execução.

Os resultados das chamadas a métodos de grupos precisam ser construídos a partir da resposta de cada membro. Do ponto de vista de quem requisitou o comando, isso deve ser transparente. Tratamos casos em que os resultados de uma chamada de grupo são novos grupos (compostos a partir de objetos retornados pelos membros), mais de um valor de retorno e respostas simples como uma *string* ou um número. Neste último caso, consideramos que todas as aplicações em um grupo estão sincronizadas (i.e. são réplicas) e portanto a resposta é o valor de retorno de apenas um membro. A decisão do tipo retornado é feita a partir do valor de retorno da primeira chamada de grupo.

Observador de Grupo

O objetivo do *Observador de Grupo* é transformar as notificações de eventos que ocorrem em uma aplicação em um único evento de grupo. Essa abstração auxilia a utilização dos grupos de aplicações como uma única entidade, não somente no envio de comandos para o grupo (como explicado na Seção anterior), mas também para o tratamento dos eventos gerados por aplicações individuais desses grupos.

Como exemplo, considere um grupo com múltiplos tocadores de vídeo. Em uma determinada apresentação, gostaríamos de receber um evento assinalando a chegada do vídeo a um determinado ponto. O *Observador de Grupo* é responsável por notificar esse evento apenas uma vez em nome de todo grupo, mesmo que cada membro individualmente envie sua própria notificação.

Para esta implementação, utilizamos uma lógica simples para a escolha de uma aplicação mestre representante do grupo. Somente as notificações vindas desse mestre são repassadas pelo grupo. Uma limitação desta solução é que interações do usuário sobre uma aplicação não-mestre não serão repassadas como notificações de grupo. Uma solução onde não existem

mestres ou escravos também é viável mas se torna complexa porque requer o tratamento de diversos detalhes particulares à cada aplicação.

4.1.3 Nó Coordenador

Diferentemente do *Nó de Apresentação e Controle*, que assume uma postura passiva em relação ao envio de comandos para outras aplicações, o *Nó Coordenador* age ativamente na execução das outras aplicações. Através de chamadas sobre uma interface de automação distribuída, o *Nó Coordenador* pode controlar aplicações individualmente ou em grupo.

Para obter informações a respeito da execução das aplicações ou de interações do usuário sobre o sistema, o *Nó Coordenador* pode observar notificações publicadas pelas aplicações e pelo *Gerente de Grupo*.

A divisão entre *Nó de Apresentação* e *Nó Coordenador* é organizacional. Devido a características bastante peculiares de cada tipo de nó, a diferenciação da nomenclatura auxilia na organização e estruturação das aplicações. Através do *Nó Coordenador* implementamos a aplicação de interpretação do documento multimídia descrito no quinto protótipo (Seção 3.5).

Observador

O *Observador* é responsável pelo recebimento de notificações que podem ser enviadas pelas aplicações ou pelo *Gerente de Grupo*. As notificações obedecem a um formato flexível que permite que sejam enviados *strings*, caracteres, números ou objetos remotos. A aplicação que recebe as notificações é responsável por interpretá-las corretamente.

Para evitar que o recebimento de notificações crie ciclos de chamadas (*deadlocks*), o envio das notificações não aguarda confirmação.

Coordenador

O *Coordenador* é a aplicação que recebe e interpreta as notificações. Estas podem conter informações sobre interação do usuário, eventos ocorridos durante a execução (como erros) ou a chegada a pontos previamente determinados durante a execução.

Diferentes formas de entrada podem guiar a lógica do *Coordenador*, seja através de um sistema de inteligência artificial, um plano de execução ou até mesmo um documento multimídia.

4.1.4 Sistema de Arquivos Distribuído

O *Servidor de Arquivos* é responsável pela transferência de conteúdo para as aplicações, sendo capaz de gerenciar múltiplas conexões simultaneamente. Tipicamente, ao executar uma nova aplicação, o *Nó de Apresentação e Controle* se comunica com o *Gerente de Grupo* para obter informações sobre qual arquivo buscar e onde encontrá-lo. Nota-se que o *Servidor de Arquivos* pode ser implementado através de um sistema de diretórios, transferência HTTP ou qualquer outra forma de transferência de arquivos.

Em trabalhos futuros, poderemos absorver resultados de trabalhos como o Aura [27], que apresenta o Coda File System [33], um sistema de arquivos distribuídos que utiliza técnicas como *caching* e *data-staging* para melhorar o seu desempenho.

4.1.5 Disparador

O *Disparador* é um serviço que deve ser constantemente executado nos dispositivos que fazem parte de uma apresentação. Ele é responsável por carregar uma aplicação e fornecer o nome do grupo da aplicação, se ela pertencer a um.

O *Nó Coordenador* encontra as referências ao *Disparador* através do *Serviço de Nomes* descrito no Capítulo 2. Essa organização foi baseada no componente *UOBHost* de Gaia [26].