SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito: 20/05/2003

Assinatura: _____

AutorE: suportando autoria evolucionária em
ambientes de captura.

Daniel Gustavo Sante

Orientadora: *Profa. Dra. Maria da Graça Campos Pimentel*

Dissertação apresentada ao Instituto de Ciências Matemáticas e de
Computação – ICMC-USP, como parte dos requisitos para
obtenção do título de Mestre em Ciências da Computação e
Matemática Computacional.

USP - São Carlos
Maio de 2003

Dedicatória

Dedico este trabalho a todas as pessoas que, como eu, acreditam que o investimento em educação pode fazer a grande diferença em um país como o Brasil.

Agradecimentos

À minha mãe Sonia, minha avó Hermenegilda e minha irmã Ana Beatriz, por sempre me apoiarem nos momentos de maior necessidade.

Ao meu pai João, que tenho certeza que está olhando por mim lá de cima.

À minha orientadora Graça Pimentel, pela paciência e pelo excelente trabalho realizado.

À Radiumsystems.com, pelo suporte oferecido.

Ao ICMC-USP, por confiar em mim e permitir que realizasse esse trabalho.

Ao meu grande amigo Pedro, pelas vezes que me ajudou distrair a cabeça tocando um som e tomando um chopp.

À minha namorada Lívia, pela compreensão demonstrada nos fins-de-semana de trabalho.

A toda minha turma da COMP96, pela amizade e companheirismo por todo esse tempo.

A todos os amigos e colegas do laboratório Intermídia, especialmente Laércio, Renan, Renato e Cláudia, pela ajuda oferecida.

Aos amigos Ramon, Roger, Carol, Danilo, Fernanda, Giselle, Ricardo, Samanta, Kenji, Flávia, Gustavo, Daniela, Taís, Andréa e outros que porventura tenha me esquecido, pelo apoio moral durante esse trabalho.

A Roger Waters, por ter vindo tocar em São Paulo.

E a todas as pessoas que contribuíram direta ou indiretamente para a realização desse trabalho.

Índice

1. INTRODUÇÃO	1
1.2. MOTIVAÇÃO	2
1.3. OBJETIVOS.....	2
1.4. RESULTADOS ALCANÇADOS	3
1.5. ESTRUTURA DA DISSERTAÇÃO	3
2. INCA-SERVE E TRABALHOS CORRELATOS	4
2.1. CONSIDERAÇÕES INICIAIS.....	4
2.2. INFORMAÇÃO MULTIMÍDIA EVOLUCIONÁRIA	4
2.3. A INFRA-ESTRUTURA INCA-SERVE	5
2.4. eCLASS.....	6
2.5. eMEET	8
2.5.1. eMeet/InCA.....	9
2.5.2. eMeet/SERVE	10
2.6. xINCA.....	13
2.7. CoTEIA / CoWEB.....	13
2.8. iCLASS.....	15
2.9. CONSIDERAÇÕES FINAIS	16
3. AUTORIA BASEADA EM CAPTURA	17
3.1. CONSIDERAÇÕES INICIAIS.....	17
3.2. TEAMSPACE	18
3.3. CONFERENCE ASSISTANT	19
3.4. LECTURE BROWSER.....	21
3.5. FLATLAND	22
3.6. DOLPHIN.....	23
3.7. AUTHORING ON THE FLY (AoF)	23
3.8. eCLASS.....	24
3.9. eMEET	26
3.10. iCLASS.....	26
3.11. SUMÁRIO	27
3.12. CONSIDERAÇÕES FINAIS	27
4. PADRÕES ABERTOS	29
4.1. CONSIDERAÇÕES INICIAIS.....	29
4.2. O PADRÃO XML	30
4.3. BANCOS DE DADOS XML NATIVOS	31
4.4. O BANCO DE DADOS XINDICE	33
4.5. O SUPORTE DA PLATAFORMA JAVA	34
4.5.1. Disponibilidade de APIs.....	34
4.5.2. JSP (JavaServer Pages).....	35
4.6. CONSIDERAÇÕES FINAIS	36

5. UM MODELO ABERTO DE SUPORTE À AUTORIA EVOLUCIONÁRIA	37
5.1. CONSIDERAÇÕES INICIAIS.....	37
5.2. O MODELO CONCEITUAL PARA GERAÇÃO AUTOMÁTICA.....	37
5.3. AUTORIA EVOLUCIONÁRIA: CONSIDERAÇÕES DE PROJETO	39
5.4. UM MODELO CONCEITUAL DE SUPORTE À AUTORIA EVOLUCIONÁRIA	40
5.5. CONSIDERAÇÕES FINAIS	43
6. O SISTEMA AUTORE	44
6.1. CONSIDERAÇÕES INICIAIS.....	44
6.2. DEFINIÇÃO DOS REQUISITOS	44
6.3. MODELAGEM CONCEITUAL	46
6.4. ARQUITETURA DO SISTEMA.....	48
6.4.1. Camada de Acesso	49
6.4.2. Módulo de Preparação	50
6.4.3. Módulo de Referência	50
6.4.4. Módulo de Expansão.....	50
6.4.5. Interface Web.....	51
6.5. IMPLEMENTAÇÃO DOS MÓDULOS	51
6.5.1. Camada de Acesso	51
6.5.2. Módulos de Autoria	55
6.5.3. Módulo de Preparação	57
6.5.4. Módulo de Referência	60
6.5.5. Módulo de Expansão.....	62
6.6. UTILIZAÇÃO EM OUTROS CONTEXTOS.....	64
6.7. CENÁRIO DE USO.....	66
6.8. CONSIDERAÇÕES FINAIS	67
7. CONCLUSÕES.....	68
7.2. CONTRIBUIÇÕES	68
7.3. TRABALHOS FUTUROS	69
7.4. CONSIDERAÇÕES FINAIS	70
REFERÊNCIAS	71
APÊNDICE A - EXEMPLOS DE DOCUMENTOS XML MANIPULADOS	76
APÊNDICE B – CÓDIGOS DAS <i>INTERFACES</i> JAVA CRIADAS.....	79
APÊNDICE C – ESTRUTURA DE ARQUIVOS DO AUTORE	83

Índice de Figuras

FIGURA 1 – ESPIRAL REPRESENTANDO O CRESCIMENTO CONTÍNUO DA INFORMAÇÃO MULTIMÍDIA EVOLUCIONÁRIA (PIMENTEL ET AL., 2001)	5
FIGURA 2 – SALA DE AULA DO AMBIENTE ECLASS, COM [A] LOUSA ELETRÔNICA, [B] CÂMERA DE VÍDEO, [C] MICROFONES, [D,E] TELAS COM IMAGENS DOS DOIS PROJETORES DE VÍDEO.	7
FIGURA 3 - HIPERDOCUMENTO GERADO AUTOMATICAMENTE PELO ECLASS APÓS O TÉRMINO DE UMA SESSÃO DE CAPTURA (SEGUNDO PLANO).....	8
FIGURA 4 – ESQUEMA ILUSTRANDO A DIFUSÃO E GRAVAÇÃO DAS MÍDIAS CAPTURADAS ENTRE OS CLIENTES DO EMEET/INCA, ATRAVÉS DO SERVIDOR DE CONTROLE DE REUNIÕES (BILA, 2002).	10
FIGURA 5 – A ARQUITETURA DO SISTEMA EMEET/SERVE (ELEUTÉRIO, 2002).	11
FIGURA 6 – TELA TÍPICA DO EMEET/SERVE APRESENTANDO HIPERDOCUMENTO GERADO AUTOMATICAMENTE A PARTIR DOS DADOS CAPTURADOS PELO EMEET/INCA.	12
FIGURA 7 – TELA TÍPICA DA CoWEB-ICMC, COM O MENU DE FUNCIONALIDADES (NO TOPO DA PÁGINA) E O CONTEÚDO EDITÁVEL DE FORMA COLABORATIVA (AO CENTRO).	14
FIGURA 8 – INTERFACE DA APRESENTAÇÃO SMIL DE UMA SESSÃO CAPTURADA PELO ICLASS (CATTELAN ET AL., 2002)	15
FIGURA 9 – RELAÇÃO ENTRE TIPOS DE AUTORIA E A ESPIRAL DA INFORMAÇÃO MULTIMÍDIA EVOLUCIONÁRIA.	17
FIGURA 10 – APRESENTAÇÃO DOS DADOS DE UMA REUNIÃO NO TEAMSPACE. (HTTP://WWW.RESEARCH.IBM.COM/TEAMSPACE/PROJECTDESCRIPTION.HTML).....	19
FIGURA 11 – TELA DO APLICATIVO DE RECUPERAÇÃO DE DADOS SOBRE UMA CONFERÊNCIA. (HTTP://WWW.INF.ETHZ.CH/VS/EVENTS/DAG2001/SLIDES/ANIND.PDF)	20
FIGURA 12 – HIPERDOCUMENTO GERADO AUTOMATICAMENTE PELO LECTURE BROWSER, CONSTITUÍDO DE ÍNDICE, BOTÕES DE NAVEGAÇÃO, VÍDEO, SLIDES E LINHA DO TEMPO (MUKHOPADHYAY & SMITH, 1999)	21
FIGURA 13 – INTERFACE DO FLATLAND, DURANTE UMA APRESENTAÇÃO (WHITE ET AL., 1998).....	22
FIGURA 14 – INTERFACE DE CRIAÇÃO DE UMA AULA NO AOFWB (MÜELLER & OTTMANN, 2000).....	24
FIGURA 15 – INTERFACE DE AUTORIA PARA ADIÇÃO DE INFORMAÇÃO TEXTUAL A UM SLIDE CAPTURADO NO SISTEMA ECLASS. NA PARTE INFERIOR, O FORMULÁRIO PARA ENTRADA DO TEXTO (PIMENTEL ET AL.,2000).	25
FIGURA 16 – INTERFACE DE CRIAÇÃO DE UMA NOVA AULA NO SISTEMA ICLASS.....	26
FIGURA 17 – EXEMPLO DE UM DOCUMENTO XML, UTILIZADO PARA DESCREVER UMA REUNIÃO.	31
FIGURA 18 – O MODELO CONCEITUAL PARA GERAÇÃO AUTOMÁTICA DE HIPERDOCUMENTOS (PIMENTEL ET AL., 2003).....	38
FIGURA 19 – O MODELO CONCEITUAL PARA O SUPORTE À AUTORIA EVOLUCIONÁRIA.	41
FIGURA 20 – MODELAGEM CONCEITUAL DO SISTEMA AUTORE.	47
FIGURA 21 – ESTRUTURA DAS COLEÇÕES DE DOCUMENTOS XML REPRESENTADO A BASE DE DADOS DO AUTORE.	48

FIGURA 22 – DIAGRAMA GERAL DE ARQUITETURA DO SISTEMA AUTORE.....	49
FIGURA 23 – DIAGRAMA DE CLASSES DA IMPLEMENTAÇÃO REFERENTE À CAMADA DE ACesso DO AUTORE (<i>PackDB</i>), MOSTRANDO OS COMPONENTES, SEUS MÉTODOS E PROPRIEDADES.	52
FIGURA 24 – ESQUEMA ILUSTRANDO A ESTRUTURA DE FUNCIONAMENTO DOS MÓDULOS DO AUTORE.	56
FIGURA 25– SCRIPT DE PROCESSAMENTO NO MÓDULO DE EXPANSÃO DO AUTORE-MEET. . .	57
FIGURA 26 – FORMULÁRIO DE CRIAÇÃO DE UMA NOVA AULA, NO AUTORE-CLASS.....	58
FIGURA 27 – FORMULÁRIO PARA ESCOLHA DOS SLIDES A SEREM REAPROVEITADOS, AO CRIAR-SE UMA NOVA AULA NO AUTORE-CLASS.....	59
FIGURA 28 – INTERFACE DO MÓDULO DE REFERÊNCIA, USANDO COMO EXEMPLO O AUTORE- MEET.....	61
FIGURA 29 – MENU DINÂMICO LISTANDO AULAS DISPONÍVEIS EM UMA TURMA, QUE OFERECE AO USUÁRIO AS OPÇÕES DE EXPANSÃO NO AUTORE-CLASS.	62
FIGURA 30 – FORMULÁRIO DE INSERÇÃO DE MÍDIAS NO MÓDULO DE EXPANSÃO DO AUTORE- MEET. À DIREITA, A LISTAGEM DAS MÍDIAS JÁ ADICIONADAS A ESTA REUNIÃO.	64
FIGURA 31 – A PÁGINA DE SELEÇÃO DE CONTEXTO DO SISTEMA AUTORE.....	65

Índice de Tabelas

TABELA 1 – A DIVISÃO DE SISTEMAS CSCW EM QUADRANTES DE ACORDO COM ELLIS ET AL.	8
TABELA 2 – COMPARAÇÃO ENTRE AS CARACTERÍSTICAS DE AUTORIA SUPORTADAS PELOS SISTEMAS.....	27

Resumo

Sistemas baseados em captura de experiências cotidianas têm sido desenvolvidos como forma de apoiar o fato de que as pessoas não conseguem assimilar todas as informações a que são expostas, pelos mais diversos fatores: interrupções, necessidade de anotação e sobrecarga cognitiva. No entanto, esses sistemas em sua maioria não oferecem suporte às funções de autoria interativa de informação – preparação a priori e modificação ou extensão a posteriori das informações – devido à ausência de um modelo de referência para esse suporte. O trabalho aqui apresentado consiste na proposta de um modelo aberto de suporte à autoria de informação multimídia evolucionária, e na implementação de um sistema aberto baseado nesse modelo que suporta a autoria interativa de informação e que pode ser integrado a sistemas de captura.

Abstract

Systems based on capturing everyday experiences have been developed as a form of supporting the fact that people do not assimilate all information they are submitted to, due to factors including interruptions, the need to taking notes and cognitive overload. However, these systems do not support interactive authoring – preparation of information a priori and modification and extension a posteriori – due to the lack of reference model to this support. The work presented here consists of the investigation of an open model to support authoring based on evolutionary multimedia information, and on the development of an open hypermedia system based on the model that supports interactive authoring of information and can be integrated into capture-based systems.

1. Introdução

Diariamente, as pessoas deparam-se com uma grande quantidade de informações necessárias e importantes para seu crescimento pessoal e profissional, sejam elas provenientes de uma reunião, de uma aula, da TV ou Internet. Para que todas essas informações sejam entendidas e assimiladas, é preciso que se mantenha a atenção. Porém, em várias dessas situações, a atenção pode ser desviada ou prejudicada pelos mais diversos motivos: interrupções, necessidade de anotação de informações, sobrecarga cognitiva (Conklin, 1987) e desorientação (Nielsen, 1993). Como resultado, o foco da atividade e a compreensão global sobre o assunto em questão podem ser prejudicados.

Levando-se em conta tais situações e o fato de que as pessoas não conseguem memorizar e assimilar todas as informações a que são submetidas em seu cotidiano, vários sistemas baseados na captura dessas informações foram desenvolvidos, principalmente no contexto educacional e de suporte a reuniões. No contexto educacional, visando a captura de áudio, vídeo e anotações durante uma aula ou apresentação tradicional, foi desenvolvido o sistema *eClass* (Abowd, 1999) (Abowd et al., 1998a,b) no Georgia Institute of Technology. O *eClass* conta com um ambiente equipado com dispositivos de captura para geração automática de hiperdocumentos multimídia, acessíveis via WWW (Brotherton et al, 1998). Uma remodelagem aberta desse sistema, denominada *iClass*, está sendo desenvolvida atualmente por pesquisadores do ICMC-USP (Cattelan et al., 2003). Outros trabalhos nesse contexto incluem o *Authoring on the Fly* (Müeller & Ottmann, 2000) e o *Lecture Browser* (Mukhopadhyay & Smith, 1999).

Já sistemas de suporte a reuniões visam apoiar as constantes reuniões realizadas em instituições ou corporações, que têm como objetivo apresentar, discutir e tomar algum tipo de decisão sobre assuntos diversos. No ICMC-USP, foi desenvolvido o sistema *eMeet* (Bila, 2002) (Eleutério, 2002) que dá suporte à captura, transmissão, apresentação, armazenamento e recuperação das informações geradas durante reuniões distribuídas. Outros trabalhos desenvolvidos nesse contexto foram os sistemas *TELEP* (Jancke et al.,

2000), *Flatland* (White et al., 1998), *DOLPHIN* (Streitz et al., 1994) e *Coral* (Minneman et al., 1995).

Em outros contextos, mas visando ainda enriquecer experiências cotidianas, existem trabalhos como o projeto *Informedia* (Wactlar et al., 1996) desenvolvido na Carnegie Mellon University, que disponibiliza bibliotecas de vídeo digital, e o *Pebbles* (Myers, 2001), que implementa interfaces para controle de PCs a partir de dispositivos móveis, como *palmtops*.

1.2. Motivação

Os sistemas de captura de informações de experiências cotidianas descritos na literatura, em sua maioria, não oferecem suporte a uma preparação prévia dos dados que serão capturados, e nem à modificação ou à inserção de novas informações a esses dados após a captura. Essas ações são denominadas funções de *autoria interativa* de informação, uma vez que as informações são preparadas ou modificadas por uma pessoa (o autor), e não somente geradas automaticamente pelos sistemas de captura.

O suporte a essas funções de autoria não é implementado em muitos sistemas devido a dificuldades associadas à autoria interativa, à falta de um modelo definido para esse suporte, e ao fato de que os sistemas são desenvolvidos em plataformas proprietárias, o que dificulta a adição de novas funcionalidades através da integração com outros sistemas.

1.3. Objetivos

Os objetivos deste trabalho são:

- Investigar um *modelo aberto de autoria* para sistemas baseados em captura das experiências cotidianas, que possa ser implementado em qualquer plataforma, e que ofereça condições para o suporte às características de autoria da informação capturada.
- Implementar um *sistema aberto* que suporte a autoria da informação capturada, baseado no modelo investigado, e que possa ser integrado aos sistemas de captura para suprir as funcionalidades de autoria necessárias.

1.4. Resultados Alcançados

Como resultado desse trabalho, foi proposto um modelo aberto de suporte à autoria evolucionária, cuja abordagem propõe resolver os principais problemas associados ao projeto de sistemas de autoria no contexto de informações capturadas, e que pode ser implementado em qualquer plataforma.

Também foi desenvolvido um sistema Web, o *AutorE*, que é um sistema baseado em padrões abertos XML, com suporte às operações de autoria e que inicialmente oferece funcionalidades para os contextos educacional (AutorE-Class) e de reuniões (AutorE-Meet).

1.5. Estrutura da Dissertação

No restante desta dissertação, o Capítulo 2 descreve o conceito de informação multimídia evolucionária, apresentando o projeto InCA-SERVE, no qual o presente trabalho está inserido; o Capítulo 3 discute alguns aspectos sobre autoria baseada em captura e faz um comparativo entre sistemas; o Capítulo 4 apresenta conceitos relacionados a padrões abertos; o Capítulo 5 apresenta a proposta de um modelo aberto de suporte à autoria evolucionária, como extensão ao modelo de captura definido para o projeto InCA-SERVE; o Capítulo 6 apresenta o sistema AutorE, e por fim, o Capítulo 7 apresenta as conclusões sobre o trabalho desenvolvido, e possíveis trabalhos futuros derivados.

2. InCA-SERVE e Trabalhos Correlatos

2.1. Considerações Iniciais

Neste capítulo são apresentados o conceito de informação multimídia evolucionária e a infra-estrutura de software na qual o presente trabalho está inserido, a InCA-SERVE. Também são apresentados projetos relacionados a essa mesma infra-estrutura e que estão inseridos no contexto do trabalho realizado.

2.2. Informação Multimídia Evolucionária

Qualquer atividade interativa periódica que envolva um grupo de pessoas (por exemplo, reuniões regulares de um grupo de trabalho) gera um conjunto de informações dinâmicas em relação a cada sessão apresentada. Ou seja, determinada informação só é completa quando se analisa o histórico de sua evolução em sessões passadas (relacionamento intrínseco). Tal característica deve ser levada em consideração quando se deseja automatizar o processo de geração de hiperdocumentos a partir da captura dessas atividades.

Além disso, deve-se considerar que as informações relacionadas a uma sessão não são geradas exclusivamente durante a sessão propriamente dita; elas começam a surgir antes do início da mesma (por exemplo, na preparação da pauta) e se desenvolvem também após o encerramento (por exemplo, na criação de uma ata).

Com base nisso, é possível identificar duas características importantes associadas a esse tipo de atividade (Pimentel et al., 2001):

- § Expansão progressiva: o repositório das informações capturadas se expande continuamente durante o tempo da aplicação; essa expansão é associada às contribuições dos participantes e podem ocorrer antes, durante ou depois de cada sessão de captura.

§ Referência intrínseca: o material gerado em qualquer das três fases (antes, durante ou depois do processo de captura) pode referenciar material gerado anteriormente.

Essas características são responsáveis pela complexidade das tarefas associadas à manipulação dessas informações; essas tarefas incluem o armazenamento, a recuperação e a apresentação dos dados gerados durante uma sessão de captura. Esse processo pode ser ilustrado através de uma espiral crescente (Figura 1), representando o crescimento contínuo do repositório de dados, e a informação resultante é chamada de *informação multimídia evolucionária*.

Figura 1 – Espiral representando o crescimento contínuo da informação multimídia evolucionária (Pimentel et al., 2001)

2.3. A Infra-estrutura InCA-SERVE

O projeto InCA-SERVE investiga os problemas associados a suportar informação multimídia evolucionária, através da construção de duas infra-estruturas complementares: InCA e SERVE.

A infra-estrutura InCA (*Infrastructure for Capture and Access*) foi construída como uma estrutura escalável, extensível e flexível, responsável pela captura e integração de atividades interativas genéricas. Além de permitir a captura de atividades realizadas durante

as sessões (ao vivo), também provê um modelo de informações extensível para a captura e armazenamento de experiências interativas, dando suporte à informação multimídia evolucionária (Truong & Abowd, 2001).

Utilizando-se dos serviços da InCA, a infra-estrutura SERVE (*Store, Extend, Retrieve, Visualize, Evolutionary*) é responsável por armazenar, estender, recuperar e apresentar a informação multimídia evolucionária. Ela deve funcionar como uma servidora das informações capturadas e deve também suportar serviços especializados externos à sessão capturada.

Vários projetos já foram realizados dentro do contexto InCA-SERVE, que estão relacionados ao trabalho de desenvolvimento de um *framework* comum de suporte à informação multimídia evolucionária. Alguns desses projetos serão apresentados nas próximas seções.

2.4. eClass

O eClass é um projeto cujo objetivo é estudar a utilização da computação ubíqua em ambientes educacionais através da captura de informações do ambiente e da produção automática de documentos hipermídia que refletem todo o conteúdo capturado, com a apresentação dos mesmos através da WWW (Brotherton et al., 1998).

O sistema é composto por infra-estruturas de hardware e de software. A infra-estrutura de hardware típica, presente em uma sala de aula instrumentada, consiste em uma lousa eletrônica, dois projetores, uma câmera de vídeo e vários microfones embutidos (Figura 2). Esse equipamento é responsável pela captura das várias mídias geradas em uma aula e opera de maneira síncrona com os módulos de software (Pimentel et al., 2001).

Figura 2 – Sala de aula do ambiente eClass, com [A] lousa eletrônica, [B] câmera de vídeo, [C] microfones, [D,E] telas com imagens dos dois projetores de vídeo (Pimentel et al., 2000).

A infra-estrutura de software, denominada ZEN* System, é um conjunto de aplicações cliente-servidor responsáveis pelo armazenamento e sincronização das *streams* de informação capturadas na sessão, além da geração dos hiperdocumentos associados ao término da mesma (Brotherton et al., 1998).

A Figura 3 apresenta um exemplo de hiperdocumento gerado ao término de uma sessão. A interface divide a tela em *frames*, onde o *frame* da direita apresenta a aula como uma sequência de slides. Cada slide corresponde às informações apresentadas pelo professor na lousa, e pode ter sido previamente preparado (por exemplo, no MS PowerPoint™) ou criado dinamicamente durante a apresentação, a partir de um slide em branco. As anotações feitas pelo professor sobre os slides (marcas ou escritas em tinta eletrônica) geram índices para o fluxo de mídia contínua (áudio ou vídeo); ao clicar sobre uma anotação, o aluno pode ouvir o áudio ou assistir o vídeo, iniciando no momento em que a anotação foi feita. Da mesma forma, as ligações existentes acima de cada slide indexam a mídia contínua pelos momentos de visitação do mesmo.

Além disso, uma interface possibilita a inserção de novos dados após a aula, como, por exemplo, anotações e comentários sobre cada slide, suportando o aspecto de expansão progressiva da informação multimídia evolucionária (Pimentel et al., 2001).

Figura 3 - Hiperdocumento gerado automaticamente pelo eClass após o término de uma sessão de captura (segundo plano). A janela em primeiro plano representa um *link* visitado durante a aula, disponível no *frame* à esquerda (Pimentel et al., 2000).

2.5. eMeet

O eMeet é um sistema CSCW (*Computer Supported Cooperative Work*) de suporte a reuniões distribuídas, que aplica vários conceitos do eClass de forma a contemplar o ambiente de uma reunião. Foi desenvolvido de forma a atender aos quatro quadrantes de sistemas CSCW propostos no *framework* de Ellis et al (Ellis et al, 1991), classificados de acordo com a separação de tempo e espaço (Tabela 1):

	Mesmo Local	Locais Diferentes
Mesmo Tempo (síncrono)	1.o Quadrante	2.o Quadrante
Tempos Diferentes (assíncrono)	3.o Quadrante	4.o Quadrante

Tabela 1 – A divisão de sistemas CSCW em quadrantes de acordo com Ellis et al.

Durante uma reunião (forma síncrona), o eMeet atende os grupos de usuário locais (1.o Quadrante) e distribuídos (2.o Quadrante). Após o término da reunião e estando o material capturado disponível na WWW (forma assíncrona), o sistema também atende grupos de usuários locais (3.o Quadrante) e distribuídos (4.o Quadrante).

O eMeet é dividido em duas partes principais: o eMeet/InCA e o eMeet/SERVE, descritas a seguir.

2.5.1. eMeet/InCA

O eMeet/InCA (Bila, 2002) é responsável pela captura e armazenamento das informações geradas durante a reunião, ou seja, áudio e vídeo dos participantes, apresentações de slides, anotações e também pela sumarização e sincronização inter-mídias capturadas.

Foi desenvolvido como um conjunto de aplicações Java, baseadas nas APIs JMF¹ e JTAPI² da Sun. Essas APIs são responsáveis pela captura das mídias de áudio e vídeo de cada participante da reunião, e pelo envio das mesmas ao servidor de controle. Esse servidor encarrega-se da difusão dessas mídias entre os outros participantes e também grava toda a sessão (Figura 4), que posteriormente poderá ser acessada através do eMeet/SERVE (descrito na próxima seção).

¹ O JMF (*Java Media Framework*) disponibiliza métodos e componentes para suporte a elementos multimídia pela linguagem Java (<http://java.sun.com/products/java-media/jmf>).

² JTAPI (*Java Telephony API*) é uma interface de programação entre aplicações de telefonia para a linguagem Java (<http://java.sun.com/products/jatapi>).

Figura 4 – Esquema ilustrando a difusão e gravação das mídias capturadas entre os clientes do eMeet/InCA, através do servidor de controle de reuniões (Bila, 2002).

O servidor de controle de reuniões foi implementado como um *daemon* (eMeetd) que aguarda requisições para o registro de participantes na reunião, através do software cliente do eMeet/InCA. Os participantes, uma vez cadastrados na sessão, enviam suas mídias de áudio e vídeo para a difusão entre os outros softwares-clientes utilizados pelos outros participantes da reunião. Essa difusão é realizada através do protocolo RTP (*Real Time Protocol*).

2.5.2. eMeet/SERVE

O eMeet/SERVE é a porção responsável pelo suporte ao acesso, recuperação e apresentação dos dados capturados através do eMeet/InCA. Também é responsável pela geração automática de hiperdocumentos a partir dos dados capturados em uma sessão (Eleutério & Pimentel, 2001). Estes hiperdocumentos são disponibilizados na WWW e tornam-se acessíveis a um grupo de trabalho específico, através de um controle de acesso aos documentos gerados de uma determinada reunião, ou a qualquer pessoa interessada, no caso de reuniões públicas.

O eMeet/SERVE foi desenvolvido baseado em um servidor Web (Apache), uma base de dados (MySQL) e uma linguagem de script (PHP) para geração dinâmica dos hiperdocumentos a partir da base de dados. Todas essas tecnologias são de utilização e distribuição gratuitas. A interface de apresentação dos hiperdocumentos para os usuários foi desenvolvida baseando-se em uma série de heurísticas de *design* de interfaces para sistemas multimídia (Martins Netto, 2002), desenvolvidas como uma especialização das heurísticas propostas por Nielsen (Nielsen, 1992).

A Figura 5 apresenta a arquitetura do eMeet/SERVE, denominada LAIAC (Eleutério, 2002). As informações armazenadas em base de dados são acessadas por uma camada independente, tornando o sistema portátil para outros SGBDs. Da mesma forma, camadas independentes tratam as informações relacionadas aos *slides* e às mídias (áudio e vídeo) provenientes de uma sessão. O acesso às informações é controlado através da utilização de um serviço de controle de usuários, e o sistema está preparado para gerar hiperdocumentos em diversas línguas, através de um conjunto de definições de idiomas.

Figura 5 – A arquitetura do sistema eMeet/SERVE (Eleutério, 2002).

Uma tela típica do eMeet/SERVE, apresentando um hiperdocumento gerado, é mostrada na Figura 6. Na parte superior direita, tem-se o menu de opções do sistema, que leva às funcionalidades de gerenciamento de usuários e salas de reuniões. Na parte central,

são apresentadas as anotações (slides) geradas durante a reunião, logo acima da linha do tempo indicando as transições entre as mesmas. Finalmente, na parte inferior esquerda são apresentados o áudio e o vídeo capturado, de forma sincronizada com as anotações.

Figura 6 – Tela típica do eMeet/SERVE apresentando hiperdocumento gerado automaticamente a partir dos dados capturados pelo eMeet/InCA.

2.6. xInCA

O xInCA (*Extended Infrastructure for Capture and Access Applications*) é um projeto desenvolvido no ICMC-USP, cuja proposta é criar um ambiente genérico, estruturado e configurável para captura e acesso de experiências do cotidiano que envolvam sessões (Cattelan, 2002). O sistema foi construído como um conjunto de módulos que provém funcionalidades básicas de captura e acesso de sessões, utilizando-se dos paradigmas da estrutura InCA desenvolvida na GeorgiaTech (Truong & Abowd, 2001).

O xInCA foi utilizado no projeto de desenvolvimento do eMeet/InCA, para a captura dos slides apresentados na sessão e também para a captura das conversas através do *chat* durante a reunião, e também no desenvolvimento do sistema iClass, que será descrito na seção 2.8.

2.7. CoTeia / CoWeb

A CoWeb é uma ferramenta CSCL (*Computer Supported Cooperative Learning*) que se destina à edição colaborativa de páginas na WWW, que podem ter conteúdo dinâmico. Através de uma série de diretivas próprias do ambiente, os usuários podem criar e editar suas páginas na WWW de maneira compartilhada. A CoWeb foi originalmente desenvolvida no Georgia Institute of Technology, em Atlanta (EUA) (Guzdial, 1999).

No ICMC-USP, foi realizada uma nova implementação da CoWeb, denominada CoTeia (Arruda Jr & Pimentel, 2001). Essa nova implementação foi modelada no contexto InCA-SERVE e mantém as características de simplicidade e autoria colaborativas da CoWeb, porém preocupando-se com uma série de deficiências identificadas na implementação original (Macedo et al., 2001), tais como:

- Deficiência no controle de concorrência;
- Dificuldade de reuso de informação;
- Ausência de gerenciamento de versões;
- Limitações de recursos de navegação.

A CoTeia utiliza-se de recursos da meta-linguagem XML³ para a definição e apresentação das páginas criadas e editadas pelos usuários, tais como XHTML⁴, DTDs⁵ e folhas de estilo XSLT⁶, que são pré-processadas para tornar a apresentação das páginas independente do *browser* Web utilizado pelo usuário.

A Figura 7 mostra uma tela típica da CoTeia. Pode-se ver o menu de opções na parte superior da tela, a partir do qual é possível editar a página atual, ver o registro de modificações, realizar *upload* de arquivos, fazer uma busca ou acessar a sala de *chat*, dentre outras funcionalidades disponíveis.

Figura 7 – Tela típica da CoTeia, com o menu de funcionalidades (no topo da página) e o conteúdo editável de forma colaborativa (ao centro).

³ XML (*Extensible Markup Language*) é um formato universal para documentos estruturados e dados (<http://www.w3.org/XML/>). Será discutido em mais detalhes no Capítulo 4.

⁴ XHTML é a reformulação da linguagem HTML como uma aplicação XML (<http://www.w3.org/TR/xhtml1>).

⁵ DTD (*Document Type Definition*) contém a definição de estrutura para validação de um documento XML.

⁶ Folhas de estilo XSLT (*Extensible Stylesheet Language – Transformation*) são utilizadas para transformar documentos XML em outros documentos, como por exemplo, documentos HTML que podem ser interpretados por qualquer *browser* Web (<http://www.w3.org/TR/xslt>).

2.8. iClass

O iClass é um sistema de captura de sessões em ambiente de sala de aula. É uma evolução em termos de estruturação de software do sistema eClass descrito na Seção 2.4, e visa substituí-lo (Cattelan et al., 2002).

O sistema iClass foi construído com base nas infra-estruturas e serviços para construção de aplicações de captura e acesso InCA (descrita na seção 2.3), xInCA (descrita na Seção 2.6), INDU (<http://catuaba.icmc.usp.br/indu>) e StRES (Baldochi Jr., 2002). Ele implementa as principais funcionalidades do sistema eClass e adiciona outras, como o suporte a dispositivos móveis.

A funcionalidade de acesso às sessões capturadas no ambiente de aulas também foi estendida em relação ao eClass. Para cada sessão, são apresentadas as opções de visualização de documentos SMIL⁷ e XHTML+SMIL, além do HTML padrão, gerados automaticamente após a captura. Os documentos SMIL podem ser reproduzidos por um *player* como o RealOne, e o XHTML+SMIL pelo *browser* Internet Explorer 5.5 ou superior. A Figura 8 mostra uma sessão sendo apresentada em SMIL com slides e linha do tempo, onde cada marcador representa um *link* para o momento de visita a um slide.

Figura 8 – Interface da apresentação SMIL de uma sessão capturada pelo iClass (Cattelan et al., 2002)

⁷ SMIL (*Synchronized Multimedia Integration Language*): padrão XML para apresentações multimídia sincronizadas (<http://www.w3.org/AudioVideo>).

2.9. Considerações Finais

A infra-estrutura InCA-SERVE já conta com vários sistemas integráveis que oferecem suporte à informação multimídia evolucionária, como mostrado nesse capítulo. No entanto, não existe um sistema de autoria que suporte a informação multimídia evolucionária durante as fases Antes da Sessão e Depois da Sessão (ver seção 2.2) e que possa ser utilizado para autoria de documentos nos diversos contextos da InCA-SERVE.

No próximo capítulo serão discutidos aspectos relacionados à Autoria baseada em Captura, apresentando a abordagem de alguns sistemas em relação a esse assunto.

3. Autoria baseada em Captura

3.1. Considerações Iniciais

Existem diversos aspectos relacionados à autoria de documentos hipermídia de forma geral; neste capítulo, será tratado especialmente o caso de autoria baseada em informações capturadas a partir de sessões ao vivo. As características analisadas incluem: autoria automática de hiperdocumentos a partir de dados capturados; autoria interativa de preparação; reutilização de material prévio; autoria de expansão (*augmenting*); suporte a elementos multimídia; geração de documentos com base em padrões abertos (XML). A relação entre os tipos de autoria apresentados e a espiral de crescimento da informação multimídia evolucionária é ilustrada na Figura 9.

Figura 9 – Relação entre tipos de autoria e a espiral da informação multimídia evolucionária. AP significa autoria de preparação; AA, autoria automática e AE, autoria de expansão.

A autoria automática é caracterizada pela geração automática de hiperdocumentos através do processamento dos dados capturados, normalmente através de sincronização em uma linha do tempo, e ocorre durante a sessão de captura. A autoria interativa de preparação engloba a criação de um documento anterior à sessão, que servirá de base para a mesma, e pode incluir ou não o suporte a reutilização de material previamente capturado. A autoria de expansão permite que se adicionem informações a documentos, após o término da sessão, permitindo a expansão progressiva da base de informações. O suporte a

elementos multimídia indica que o sistema de autoria inclui suporte a elementos de áudio e vídeo. A geração de documentos XML, ao contrário de abordagens proprietárias, facilita a extensão dos mesmos, bem como a interoperabilidade entre os diversos sistemas.

Neste capítulo, são apresentados alguns sistemas baseados em captura de informação, tanto no contexto educacional quanto no contexto de reuniões, e analisadas as características de autoria que cada um deles suporta. Esses sistemas foram escolhidos para estudo por serem sistemas representativos no conjunto de ambientes de captura, com muitas referências na literatura.

3.2. *TeamSpace*

O TeamSpace (Richter et al., 2001) é um *workspace* colaborativo para grupos que gerencia processos de trabalho compartilhados e mantém artigos em comum no caso de projetos distribuídos. Os objetivos do TeamSpace são suportar a colaboração entre companhias através de alertas, compartilhamento de informações, comunicação e coordenação, tanto nas atividades de grupo síncronas como nas assíncronas. Uma de suas funcionalidades é o suporte a atividades relacionadas a reuniões, o que inclui a preparação de novas reuniões, captura dos dados da reunião e posterior acesso a esses dados.

A preparação de uma reunião é feita agendando-se uma sala de reunião virtual, adicionando-se participantes (registrados no sistema) a ela, definindo os itens a serem discutindo na reunião, e inserindo apresentações (slides) que serão exibidas durante a reunião. O sistema de preparação não inclui mecanismo de autoria prévia baseada em informação capturada anteriormente (autoria de preparação).

Durante a reunião, os participantes podem visualizar, editar e anotar suas agendas e itens de discussão, bem como visualizar e anotar sobre os slides apresentados, e esses eventos são todos capturados juntamente com o áudio e vídeo da reunião. Após o término da sessão, os dados são automaticamente disponibilizados através da geração de um hiperdocumento segmentado contendo linhas do tempo, que provê um resumo visual dos eventos da reunião. Os dados desse hiperdocumento (apresentações, agenda, itens de discussão e áudio/vídeo) são provenientes de documentos XML armazenados localmente no sistema de arquivos. A sessão é apresentada então por um *applet* na forma de painéis

separados ou janelas independentes, para melhor organização, mostrados na Figura 10. Não é possível a adição de nova informação à sessão capturada (autoria de expansão).

Figura 10 – Apresentação dos dados de uma reunião no TeamSpace. Na janela maior, é mostrado o *applet* de visualização contendo as linhas do tempo (no alto) e os painéis segmentados com o conteúdo do hipertexto gerado (ao centro), que pode ser apresentado no próprio painel ou em janelas independentes, mostradas à esquerda (<http://www.research.ibm.com/teamspace/projectdescription.html>).

3.3. Conference Assistant

O Conference Assistant (Dey et al., 1999) é uma aplicação sensível a contexto, projetada para ser usada por participantes de conferências, onde normalmente há vários eventos simultâneos como apresentações de *papers*, demonstrações e reuniões temáticas de grupos de trabalho. Utiliza elementos de computação móvel (*wearable computing*) e um ambiente-protótipo para identificar os contextos de tempo, identidade, localização e atividade.

Durante uma conferência, o sistema apresenta uma série de informações relacionadas ao evento que se está assistindo no momento e permite que o utilizador tome

notas a respeito de slides e páginas Web apresentadas. O usuário pode a qualquer momento dirigir-se a outro evento (palestra da conferência, por exemplo), e o contexto é alterado automaticamente para as novas informações.

Após o fim da conferência, é possível recuperar os dados referentes a uma determinada apresentação (áudio, vídeo, anotações, páginas Web etc.), que são disponibilizados através de geração automática a partir dos dados colhidos durante a conferência. O sistema, mostrado na Figura 11, apresenta uma linha do tempo e um mecanismo de busca baseado em contexto, ou seja, a linha do tempo pode indicar eventos como a entrada e saída de uma pessoa de uma sala, momentos em que foram feitas perguntas, ou quando uma determinada palavra foi utilizada. Selecionando um evento na linha do tempo, o usuário pode ver o slide ou página Web apresentado no momento daquele evento, o áudio e vídeo gravados durante a apresentação do slide e quaisquer anotações pessoais que porventura tenham sido feitas sobre a informação apresentada.

Figura 11 – Tela do aplicativo de recuperação de dados sobre uma conferência. No alto, são mostradas a interface de busca e a linha do tempo; na parte inferior, os lides apresentados e informações de anotações (<http://www.inf.ethz.ch/vs/events/dag2001/slides/anind.pdf>).

Apesar de utilizar XML na comunicação entre os dispositivos móveis e os servidores de contexto, os dados referentes às apresentações capturadas no Conference Assistant são armazenados em um *Presentation Server* proprietário.

3.4. Lecture Browser

O Lecture Browser (Mukhopadhyay & Smith, 1999) é um sistema de captura e apresentação de aulas, seminários, palestras e outros tipos de apresentação, que também gera documentos disponibilizados na Web. O sistema é constituído por duas partes principais: o software de captura, que recebe as *streams* de vídeo e os dados da apresentação para sincronizá-los, e o servidor de processamento, que recebe os dados sincronizados e gera a apresentação (em formato HTML) que posteriormente poderá ser visualizada através de um *browser* Web. A Figura 12 apresenta um documento disponibilizado na Web pelo Lecture Browser.

Figura 12 – Hiperdocumento gerado automaticamente pelo Lecture Browser, constituído de índice, botões de navegação, vídeo, slides e linha do tempo (Mukhopadhyay & Smith, 1999).

O objetivo (e também a principal característica) do Lecture Browser é a automatização do processo de autoria de hiperdocumentos a partir de eventos ao vivo, em ambientes estruturados, gerando hiperdocumentos também estruturados. Dessa forma, não há suporte a outras formas de autoria (de preparação e expansão) baseada nas informações capturadas pelo sistema.

3.5. Flatland

O Flatland (White et al., 1998) é um sistema de suporte a educação à distância *desktop-to-desktop*, ou seja, não é baseado na captura de informações multimídias em aulas (ou apresentações) presenciais, e sim em tele-aulas. É uma aplicação cliente-servidor, distribuída, que permite vários tipos de interação entre os participantes conectados, como, por exemplo, visualizar e ouvir o professor, discussões entre os participantes, envio de perguntas e outras interações típicas de uma aula ou palestra.

A principal característica do Flatland é permitir a pré-autoria da estrutura da aula, através da definição da configuração dos elementos que serão vistos na interface de cada aluno (área de vídeo, área de slides e questões, área de discussão, etc.) além da definição do próprio conteúdo da aula (questões que serão propostas aos alunos, slides que serão apresentados etc.). A Figura 13 mostra a interface do Flatland durante uma apresentação.

Figura 13 – Interface do Flatland, durante uma apresentação (White et al., 1998).

3.6. Dolphin

O Dolphin (Streitz et al., 1994) é uma aplicação *groupware* que foi desenvolvida para suportar diversos tipos de reuniões: reuniões face-a-face, apoiadas por uma lousa eletrônica, com ou sem computadores disponíveis para os participantes; extensões dessas reuniões, com participantes remotos conectados via computadores e redes de áudio e vídeo; e participantes em uma segunda sala também contendo lousa eletrônica e/ou computadores.

Para oferecer a maior liberdade possível ao usuário para criar diferentes tipos de documentos (informações) e também para a criação de relacionamentos explícitos entre eles (ou parte deles), foi adotada uma abordagem hipermídia generalizada no desenvolvimento do sistema. Todos os objetos e relacionamentos são tratados como objetos hipermídia e possuem *links* em vários níveis de complexidade (por exemplo, objetos compostos em camadas ou embutidos). O modelo de documento do Dolphin é composto por: nós, conteúdo dos nós, ligações, anotações na lousa eletrônica (*handwriting*), texto e imagem. Esse modelo de documento provê a estrutura necessária para que os usuários possam realizar as tarefas típicas antes, durante e após a reunião.

O Dolphin não captura nem sincroniza elementos de áudio/vídeo com os hiperdocumentos gerados durante a reunião. Porém, o sistema permite que o material criado em uma reunião seja reaproveitado com base para outros documentos, em um processo colaborativo de edição posterior. Da mesma forma, documentos criados antes ou depois da reunião podem ser reutilizados em outras reuniões subseqüentes. O Dolphin não é um sistema baseado na Web, ou seja, seus hiperdocumentos (proprietários) não são visualizados através de um *browser* Web e sim por um módulo do próprio sistema.

3.7. Authoring on the Fly (AoF)

O *Authoring on the Fly (AoF)* (Müeller & Ottmann, 2000) é um sistema usado para gravação de aulas e eventos educacionais ou de treinamento. As mídias capturadas incluem slides, anotações do professor, áudio, vídeo e comandos de controle do próprio sistema. Após a captura das sessões, todo o material capturado é processado e são gerados automaticamente os hiperdocumentos que permitem a reprodução das mesmas, através do modelo de sincronização do *AoF*.

Para a preparação das sessões a serem capturadas, existe um módulo de autoria (*AOFwb*) que integra um editor de textos e gráficos. Algumas funcionalidades desse editor são a importação de arquivos de imagens em vários formatos, formatação de texto e funções de *clipboard*. Um exemplo da interface de preparação do *AOFwb* é mostrado na Figura 14.

Figura 14 – Interface de criação de uma aula no *AOFwb* (Müeller & Ottmann, 2000)

Após a aula é possível editar as sessões capturadas para corrigir erros de digitação ou eliminar trechos de áudio com problemas, através de um módulo de edição dos documentos gerados pelo *AoF*, porém não existe o conceito de autoria de expansão dentro do contexto de informação multimídia evolucionária. Esses documentos gerados pelo *AoF* são arquivos-texto contendo a lista de objetos sincronizados e a fila de comandos, em formato proprietário.

3.8. *eClass*

Existem duas formas de autoria de hiperdocumentos no sistema *eClass* (Brotherton et al., 1998): a geração automática e a autoria de expansão, ambas fundamentadas numa estrutura de hipertexto pré-definida, baseada em nós e operações de ligação sobre esses nós. Essas formas de autoria serão discutidas a seguir.

O sistema eClass captura várias *streams* de informação durante uma sessão, e cada *stream* é tratada como uma sequência individual de eventos associados a um *timestamp*. O processamento dessas *streams*, de acordo com uma escala de tempo, permite a geração automática de estruturas lineares baseadas no tempo que correspondem a estruturas de hipertexto associadas a documentos contendo essas *streams*, que podem ser contínuas (áudio/vídeo) ou discretas (páginas Web visitadas, slides, tinta eletrônica). Cada um desses elementos corresponde a um nó na estrutura de hipertexto do eClass.

O suporte à expansão da informação capturada é implementado de duas formas: a primeira permite que a informação capturada durante a sessão seja interativamente expandida, através da adição de informação textual e de anotações; já a segunda permite que usuários interativamente integrem informações produzidas fora da sessão de captura ao corpo da informação hipermídia gerada (para isso, é utilizada a CoWeb). Isso é possível já que a estrutura-base do hipertexto é armazenada de forma independente do conteúdo da informação capturada. A Figura 15 apresenta a interface para que se adicione informação textual a um slide, através de um formulário.

Figura 15 – Interface de autoria para adição de informação textual a um slide capturado no sistema eClass. Na parte inferior, o formulário para entrada do texto (Pimentel et al.,2000).

3.9. eMeet

O sistema eMeet (Bila, 2002) (Eleutério, 2002) suporta apenas a geração automática de hiperdocumentos, com estrutura similar aos hiperdocumentos gerados pelo sistema eClass. São capturadas várias *streams* de informação durante a reunião, que são processadas e sincronizadas de acordo com uma linha de tempo, sendo então gerado um hiperdocumento HTML a partir dos dados armazenados na base de dados (relacional) e que é disponibilizado via Web.

Atualmente, não há suporte às formas de autoria de preparação e expansão, bem como não é possível a reutilização em novas sessões de material gerado anteriormente.

3.10. iClass

O iClass (Cattelan et al., 2002) estendeu parcialmente as funcionalidades de autoria herdadas do eClass, que é o sistema do qual foi derivado. Por exemplo, foi implementado o suporte à preparação prévia das aulas a serem capturadas com o *upload* de mídias (no caso, slides). A Figura 16 mostra a interface de criação de uma aula.

The screenshot shows a web browser window titled "Bem-vindo ao sistema iClass - Microsoft Internet Explorer". The address bar shows the URL "C:\Meus documentos\USP\Bem-vindo ao sistema iClass.htm". The page features the "iClass" logo and logos for "Powered by InCA", "Powered by XINCA", "inDU", and a blue knot logo. Below the logos, there are dropdown menus for "Cursos" (set to "Hipermedia (SCE-5811)") and "Período" (set to "1o Semestre" and "2003"). Under the "Aula" section, there are two "Criar Nova Aula" buttons, a "Título" text field (containing "Nova Aula"), and a "Mês/Dia/Ano" date selector (set to "Apr", "25", "2003"). A checkbox labeled "Upload Slides" is checked. Below these are input fields for "Usuário" (containing "admin") and "Senha" (containing "*****"). At the bottom, there are three buttons: "Iniciar Aula", "Assistir Aula", and "Criar Aula". A link "Repositório de aulas" is also visible.

Figura 16 – Interface de criação de uma nova aula no sistema iClass.

Da mesma forma que no eClass, hiperdocumentos são gerados automaticamente com a captura de sessões. No entanto, em relação à geração automática, houve uma evolução no sentido de que as sessões capturadas são representadas por documentos XML e armazenadas em uma base aberta. Por outro lado, a funcionalidade de expansão das informações capturadas não foi implementada. O iClass também não suporta o reuso das informações capturadas anteriormente ou de material utilizado na preparação de aulas anteriores para a criação de novas aulas.

3.11. Sumário

A Tabela 2 sumariza as principais características de autoria suportadas pelos sistemas apresentados.

	Geração automática	Autoria de preparação	Autoria de expansão	Reuso de informação	Elementos multimídia	Base aberta
<i>TeamSpace</i>	sim	sim	não	não	sim	sim
<i>Conference Assistant</i>	sim	não	sim	não	sim	não
<i>Lecture Browser</i>	sim	não	não	não	sim	não
<i>Flatland</i>	não	sim	não	não	sim	não
<i>Dolphin</i>	sim	sim	sim	sim	não	não
<i>AoF</i>	sim	sim	não	não	sim	não
<i>eClass</i>	sim	não	sim	não	sim	não
<i>eMeet</i>	sim	não	não	não	sim	não
<i>iClass</i>	sim	sim	não	não	sim	sim

Tabela 2 – Comparação entre as características de autoria suportadas pelos sistemas.

3.12. Considerações Finais

Como apresentado nesse capítulo, existem várias características relacionadas à autoria e suporte à expansão da informação, bem como reutilização de material anterior, que não são suportados pelos sistemas apresentados. A utilização de padrões abertos para armazenamento dos dados relativos às informações capturadas também não é uma prática comum à maioria desses sistemas.

A sumarização das características de autoria dos sistemas estudados, apresentada na Tabela 2, serviu como base para a definição de requisitos do sistema AutorE e das características desejáveis de um sistema de suporte à autoria baseado em informação evolucionária.

No próximo capítulo serão discutidos com mais detalhes os conceitos sobre padrões abertos, apresentando o XML e sua relação bancos de dados, além das características de suporte da plataforma Java que levaram à escolha da mesma para o desenvolvimento desse trabalho.

4. Padrões Abertos

4.1. Considerações Iniciais

Segundo (Perens, 2003), padrões abertos são mais que especificações. Os princípios por trás dos padrões e a prática do oferecimento e da operação do padrão é que o torna um padrão aberto. Esses princípios são definidos por Perens como sendo:

- Disponibilidade. Os padrões abertos estão disponíveis para que qualquer um possa estudá-los e implementá-los.
- Poder de escolha. Os padrões abertos não forçam as pessoas a escolherem uma determinada implementação de um grupo em particular.
- Sem *royalties*. Não é necessário o pagamento de *royalties* para a utilização ou implementação de um padrão aberto. No entanto, a certificação de conformidade com o padrão pode ser cobrada.
- Sem discriminação. Apenas critérios técnicos de adequação ao padrão são levados em conta para determinar o favorecimento de uma implementação específica.
- Extensão ou Subconjunto. Implementações de padrões abertos podem ser apresentadas na forma de extensões ou subconjuntos dos padrões, porém podem ser exigidos alguns requisitos extras para a certificação da conformidade com o padrão.
- Sem práticas predatórias. Os padrões abertos devem incluir em sua licença de uso termos que impeçam a subversão dos mesmos.

A seguir, será apresentado brevemente o padrão aberto XML e também feita uma discussão sobre a relação entre XML e bancos de dados, apresentando um banco de dados XML nativo (o Xindice). Por fim, será discutido o suporte da plataforma Java ao padrão aberto XML.

4.2. O Padrão XML

XML (*Extensible Markup Language*) é uma linguagem de marcação, apropriada à representação de dados, documentos e outras entidades, cuja essência é fundamentada na capacidade de agregar informações (Pimentel & Teixeira, 2000). Exemplos de entidades são programas de computador, *scripts* ou folha de estilos. O modelo de *markup*, onde marcas específicas são utilizadas para identificar os dados contidos em um documento, permite que XML seja uma linguagem aberta, *human-readable* e capaz de manter independentes as representações de estrutura, conteúdo e semântica dos documentos.

O padrão XML estabelece regras gerais às quais documentos em conformidade com XML devem respeitar. Além disso, XML é uma meta-linguagem, ou seja, provê mecanismos para a definição de novas linguagens para classes de documento específicas, com estrutura, conjunto de elementos (*tags*) e atributos bem definidos. Isso permite que os autores de documentos criem a melhor definição para seus documentos, de acordo com seu interesse, permitindo que aplicações associem semântica aos dados do documento e viabilizando o processamento automático dos mesmos.

Um exemplo de documento XML utilizado para descrever uma reunião é mostrado na Figura 17. Os elementos são delimitados pelos sinais `<` e `>` que compõem uma *tag* XML. Elementos podem conter outros elementos ou conteúdo (texto) propriamente dito. A cada *tag* de abertura de elemento (por exemplo, `<data>`) deve corresponder uma *tag* de fechamento (`</data>`), ou no caso de elementos sem conteúdo, o fechamento pode vir no final da própria *tag* (por exemplo, `<sessoes/>`). Os atributos são associados aos elementos, caracterizando-os, e devem ter valores definidos (por exemplo, `id="0"`). Na primeira linha, existe uma *instrução de processamento* determinando a versão da especificação e o conjunto de caracteres utilizado pelo documento XML (`<?xml version="1.0" encoding="UTF-8"?>`). O primeiro elemento de um documento XML é denominado a *raiz* do documento (no exemplo, `<reuniao>`).


```

<?xml version="1.0" encoding="UTF-8"?>
<reuniao id="0">
  <titulo>Próximos Passos</titulo>
  <resumo>Definição das próximas etapas do projeto</resumo>
  <data>12/12/2003</data>
  <acesso>0</acesso>
  <pessoas>
    <pessoa id="João" papel="participante"/>
    <pessoa id="Carlos" papel="secretário"/>
  </pessoas>
  <pauta>
    <item id="0" descricao="Definição dos passos"/>
  </pauta>
  <bg_slides>
    <bg_slide id="0" href="http://localhost:8080/slide0.jpg"/>
    <bg_slide id="1" href="http://localhost:8080/slide1.jpg"/>
    <bg_slide id="2" href="http://localhost:8080/slide2.jpg"/>
    <bg_slide id="3" href="http://localhost:8080/slide3.jpg"/>
  </bg_slides>
  <sessoes/>
</reuniao>

```

Figura 17 – Exemplo de um documento XML, utilizado para descrever uma reunião.

Os documentos que respeitam todas as regras de sintaxe da recomendação XML são denominados documentos XML bem-formados. Documentos que não sejam bem-formados não podem ser processados por aplicações XML e nem apresentados por *browsers*.

4.3. Bancos de Dados XML Nativos

Com a disseminação do padrão XML, surgiu a oportunidade de uma nova forma de armazenagem centrada em documentos. Os modelos de bancos de dados tradicionais usualmente armazenam dados, e o suporte à armazenagem de documentos é obtido através de um mapeamento prévio do esquema do documento em uma estrutura relacional, orientada a objetos ou hierárquica. Esse mapeamento geralmente impossibilita que um documento seja recuperado exatamente da mesma forma que foi armazenado, e não permite acesso direto ao documento para a manipulação de suas partes através de sua linguagem de manipulação de dados (Baldochi Jr., 2002).

Os bancos de dados XML nativos são sistemas gerenciadores de banco de dados (SGBDs) projetados especialmente para armazenar documentos XML (Bourret, 2003). Eles suportam transações, acesso multi-usuário, APIs para programação, linguagens de consulta e outras características comuns aos outros SGBDs. A diferença surge em relação ao modelo interno dos bancos XML nativos, que é baseado exclusivamente em XML e não depende de um modelo relacional. Uma definição elaborada pelos participantes da *XML:DB Mailing List*⁸ diz que um banco de dados XML nativo é aquele que:

- Define um modelo lógico para um documento XML e utiliza esse modelo para armazenar e recuperar os documentos. Exemplos de modelos são o Xpath Data Model⁹, o XML Infoset¹⁰ e o DOM¹¹.
- Tem como unidade fundamental de armazenamento um documento XML;
- Não se restringe a um modelo físico de armazenagem particular, ou seja, pode ser construído sobre outros bancos de dados, sistema de arquivos ou outro formato qualquer.

Segundo (Bourret, 2003), algumas das principais características dos sistemas de banco de dados XML nativos são:

- Suporte a coleções de documentos, definindo uma hierarquia similar a de diretórios em um sistema de arquivos para a armazenagem dos documentos.
- Linguagens de consulta apropriadas para documentos XML, como por exemplo, XPath e XQuery¹²;
- Formas de atualização e remoção de documentos ou partes de documentos, por exemplo, através da linguagem XUpdate¹³;

⁸ *XML:DB Mailing List* é uma lista de discussão para a comunidade desenvolvedora e pesquisadora de bancos de dados XML Nativos. Seus arquivos se encontram em <http://archive.xmldb.org>.

⁹ XPath define um modelo e uma linguagem de localização de elementos baseado em caminhos (<http://www.w3.org/TR/xpath>).

¹⁰ O XML Infoset provê um conjunto de definições para referências a informações em um documento XML (<http://www.w3.org/TR/xml-infoset>).

¹¹ O DOM (Document Object Model) define um modelo estrutural hierárquico para documentos XML (<http://www.w3.org/TR/DOM-Level-2-Core>).

¹² XQuery é uma linguagem de consulta que estende a funcionalidade do XPath (<http://www.w3.org/TR/xquery/>).

¹³ A especificação XUpdate encontra-se em <http://www.xmldb.org/xupdate>.

- Transações, *locking* de recursos e acesso concorrente;
- APIs disponíveis para a escrita de programas que realizem o acesso ao banco de dados, como por exemplo a API aberta *XML:DB*¹⁴.
- *Round-Tripping*, ou seja, documentos são recuperados exatamente da mesma forma que foram armazenados;
- Possibilidade de armazenar dados remotos, provenientes de outras bases de dados, junto aos documentos XML;
- Indexação de valores de elementos e atributos para melhor performance em buscas;
- Política de tratamento de referências a entidades externas; podem ser expandidas ou armazenadas junto aos documentos, ou mantidas separadamente.

As principais limitações dos bancos de dados XML nativos dizem respeito aos conceitos conhecidos de bancos de dados relacionais como normalização (evitar redundância de dados) e integridade referencial (assegurar que as ligações entre documentos sejam consistentes). Nesses casos, as aplicações que utilizam bancos XML nativos devem preocupar-se em manter a normalização e a integridade dos documentos armazenados.

4.4. O Banco de Dados Xindice

O Xindice (<http://xml.apache.org/xindice>) é um banco de dados XML nativo, originado a partir da continuação de um projeto denominado anteriormente *dbXML Core* e que foi doado à *Apache Software Foundation* em dezembro de 2001, tornando-se um projeto *Open Source*. Implementado na linguagem Java, encontra-se atualmente na versão 1.0 (estável) e pode ser instalado em qualquer sistema UNIX ou Windows.

O Xindice provê uma implementação da API aberta *XML:DB* para ser utilizada pelos desenvolvedores de aplicações Java que desejem utiliza-lo como base de dados, mas também é possível o acesso por aplicações em outra linguagens através de um *plugin* XML-

¹⁴ A API XML:DB encontra-se em <http://www.xmldb.org/xapi>.

RPC¹⁵. Além disso, já existem vários *front-ends* gráficos para administração da base de dados, como o XMLdbGui (<http://titanium.dstc.edu.au/xml/xmldbgui/>).

Como linguagem de consulta, o Xindice utiliza o XPath. A linguagem de manipulação de dados implementada no Xindice é a XUpdate. De acordo com informações no site da Apache, as próximas versões do Xindice devem implementar um *engine* XQuery como mecanismo de consulta, o que virá a suprir algumas das deficiências do XPath, como por exemplo consultas em referências cruzadas.

4.5. O suporte da plataforma Java

Java foi a plataforma escolhida para o desenvolvimento desse trabalho principalmente pelo suporte oferecido aos padrões abertos XML, através da disponibilização de APIs para manipulação de documentos e acesso a base de dados. Além disso, Java inclui uma tecnologia para a construção de aplicações Web com geração de conteúdo dinâmico, o JSP. A seguir, serão apresentadas essas características.

4.5.1. Disponibilidade de APIs

A base de dados XML nativa Xindice apresentada na Seção 4.4 oferece um implementação da API *XML:DB* em Java. Essa API oferece um conjunto de métodos que permitem a manipulação de coleções de documentos armazenados na base de dados inserindo, removendo e recuperando documentos inteiros, além de implementar os mecanismos de consulta (*query*) e atualização (*update*) dos dados contidos nos documentos.

Apesar do Xindice permitir chamadas remotas às funções da *XML:DB* através do protocolo XML-RPC por sistemas escritos em qualquer linguagem, a implementação de um sistema em Java que utilize o Xindice através de chamadas diretas às funções da API elimina o *overhead* de montagem, empacotamento, transmissão, desmontagem e recebimento dos resultados das mensagens XML-RPC.

¹⁵ XML-RPC (XML Remote Procedure Call) é um protocolo padrão baseado na troca de mensagens XML que permite que softwares rodando em diversas plataformas realizem chamadas remotas a procedimentos, através da Internet (<http://www.xmlrpc.com/spec>).

Além da API *XML:DB*, Java também conta com uma API para o processamento de documentos XML em memória – a JAXP¹⁶ (*Java API for XML Processing*) que dentre outras funcionalidades, provê um processador DOM completo que permite a criação de um novo documento, adição e remoção de nós, recuperação de atributos, etc. Os documentos criados em memória através do DOM podem ser armazenados diretamente no Xíndice, através de chamadas a funções da *XML:DB*. Da mesma forma, documentos ou trechos de documentos podem ser recuperados diretamente da base de dados como uma estrutura DOM e manipulados em memória.

4.5.2. JSP (JavaServer Pages)

A tecnologia JSP permite a construção de aplicações Web dinâmicas através da integração de código Java – os *scriptlets* – em páginas HTML ou documentos XML. Sua arquitetura permite a utilização de componentes JavaBeans¹⁷ para processamento da lógica das aplicações, e ainda apresenta uma série de *tags* especiais e funcionalidades voltadas ao desenvolvimento Web, como redirecionamento de páginas e tratamento de sessões.

Uma vantagem do JSP é a independência de sistema operacional e de servidor Web. Os sistemas JSP rodam dentro de um *servlet container*, que é um pré-processador de páginas dinâmicas. Uma implementação de referência para um *servlet container* JSP é o Tomcat, da Apache Software Foundation (<http://jakarta.apache.org/tomcat>).

Além disso, o JSP oferece a vantagem de permitir a utilização de todas as APIs disponíveis na linguagem Java para a construção de aplicações Web, o que inclui as APIs de suporte a XML e banco de dados XML nativos apresentadas na seção 4.5.1.

A utilização de APIs Java em aplicações Web não é restrita ao JSP. Outras plataformas de desenvolvimento *server-side*, como o PHP¹⁸, oferecem suporte à utilização de APIs Java, porém esse suporte é limitado e experimental. Com isso, o JSP se apresenta

¹⁶ Disponível em <http://java.sun.com/xml/jaxp>.

¹⁷ JavaBeans é uma arquitetura para a utilização de componentes reusáveis na plataforma Java. (<http://java.sun.com/products/javabeans/>)

¹⁸ PHP (Hypertext Preprocessor) é uma plataforma *open source* de construção de aplicações Web dinâmicas. <http://www.php.net>

como a escolha mais indicada para a construção de aplicações Web baseadas em componentes Java, por ser um ambiente nativo.

4.6. Considerações Finais

A escolha de padrões abertos para o desenvolvimento de sistemas traz entre outras vantagens, a facilidade de manutenção e a interoperabilidade de dados, facilitando a integração. O padrão XML é apropriado para a representação de documentos, e por ser um padrão aberto, se disseminou rapidamente com o crescimento da Internet e com a necessidade de armazenamento de informações estruturadas.

Os sistemas desenvolvidos no contexto do projeto InCA-SERVE vêm adotando o XML como padrão de dados, e Java como plataforma de desenvolvimento, tornando-os sistemas integráveis e extensíveis.

No próximo capítulo, será proposto um modelo aberto de suporte à autoria de informação multimídia evolucionária, como uma extensão ao modelo existente no contexto InCA-SERVE.

5. Um Modelo Aberto de Suporte à Autoria Evolucionária

5.1. Considerações Iniciais

De acordo com a tabela apresentada na Seção 3.11, vemos que a maioria dos sistemas de captura não oferece o suporte adequado às características de autoria da informação multimídia evolucionária. O projeto InCA-SERVE não tem um modelo bem definido para o suporte a essas características.

A seguir, será apresentado um modelo de suporte à geração automática de hiperdocumentos, suas deficiências e alguns problemas em relação aos aspectos de autoria. Por fim, será definido, como extensão a esse modelo, um modelo aberto de suporte à autoria evolucionária.

5.2. O Modelo Conceitual para Geração Automática

Segundo (Pimentel et al., 2003), os sistemas de geração automática de hiperdocumentos a partir de informação capturada, em ambientes de computação ubíqua, apresentam dois problemas que devem ser abordados em tempo de projeto, para que esses sistemas possam ser desenvolvidos corretamente e cumpram as funções desejadas:

- Caminho de Interação: toda a interação do usuário com os dispositivos de um sistema de captura (câmeras, microfones, *browsers*) deve estar representada na informação capturada, exatamente na mesma ordem em que ocorreu. Isso é necessário para que posteriormente, quando os dados capturados forem recuperados para apresentação, esses caminhos de interação sejam representados por *links* nos documentos hipertexto.
- Dados de Primeira Classe: Os dados capturados devem ser tratados como objetos de primeira classe, independentes, permitindo uma estruturação dos hiperdocumentos gerados de acordo com as classes de objetos capturados, o que permite o suporte às operações necessárias para a apresentação de cada objeto (por exemplo, uma *stream* de áudio ou uma imagem).

Para a solução desses problemas, um modelo conceitual para geração automática de hiperdocumentos a partir de dados provenientes de captura é apresentado por Pimentel et al. (2003). Esse modelo é mostrado na Figura 18. Ele foi proposto com o objetivo de identificar os principais cenários de uso baseados na disponibilidade de dispositivos de captura (identificando as classes de dados) e nos relacionamentos presenciados em um evento de captura ao vivo, que devem ser disponibilizados para futura consulta (identificando o caminho de interação).

Figura 18 – O modelo conceitual para geração automática de hiperdocumentos (Pimentel et al., 2003).

Apesar do modelo proposto servir bem à geração automática de hiperdocumentos, ele é um modelo estático, ou seja, não prevê a evolução da informação capturada. A autoria evolucionária da informação apresenta outras considerações não previstas nesse modelo, e que serão discutidas na próxima seção.

5.3. *Autoria Evolucionária: Considerações de Projeto*

O processo de suporte à autoria da informação multimídia evolucionária deve levar em consideração alguns problemas identificados para o tratamento da informação, que influenciam o projeto dos sistemas de autoria. Esses problemas são os seguintes:

- O Problema de Contextos. Os dados que compõem a informação de autoria variam entre os contextos dos ambientes de captura em vários aspectos, como terminologia, semântica, e tipo. Por exemplo, no contexto de uma aula, são comuns os termos “professor” e “aluno” para designar os participantes da mesma, e quem normalmente realiza as ações pode ser principalmente o professor; isso não ocorre no contexto de uma reunião de grupo de trabalho, onde há o “presidente”, o “secretário” e os demais “participantes”, e todos atuam ativamente na reunião. Um resumo feito ao término de uma aula tem um significado semântico diferente da ata de uma reunião, apesar de apresentarem o mesmo tipo (texto). O mesmo se aplica a outros contextos; apesar de possuírem características comuns, não existe um conjunto de dados genérico que consiga representar todos os contextos de quaisquer ambientes de captura.
- O Problema da Localização. A informação de autoria é proveniente e deve estar localizada em algum lugar, *logicamente*, de maneira estruturada que permita sua classificação e independente de sua localização física. Essa localização se dá em diversos níveis e também é dependente de contexto. Por exemplo, uma aula normalmente é dada em uma sala de aula (física) dentro de uma escola ou universidade, mas também está inserida logicamente dentro de uma *turma* de alunos, que cursam uma *disciplina*, que compõem um *curso*. Da mesma forma,

uma reunião pode ser realizada em uma mesma *sala* lógica, mesmo que esteja distribuída geograficamente em diversas salas.

- O Problema de Reuso. Deve ser possível reutilizar informação previamente inserida no processo de autoria, evitando trabalho desnecessário e facilitando ao máximo o processo aos autores, que normalmente já são submetidos a grande sobrecarga cognitiva no processo de autoria de hiperdocumentos em geral (Conklin, 1987).

5.4. *Um Modelo Conceitual de Suporte à Autoria Evolucionária*

Um modelo conceitual para o suporte à autoria de informação multimídia evolucionária, definido como uma extensão ao modelo de geração automática apresentado na Seção 5.2, é mostrado na Figura 19. Esse modelo foi proposto para unificar o tratamento dos contextos de autoria (que diz respeito ao Problema dos Contextos), para oferecer mecanismos estruturados em relação à localização das informações (Problema da Localização) e apresentar componentes independentes de informação que possam ser reutilizados (Problema do Reuso).

A classe *Session* foi definida como ponto de partida do modelo estendido; ela é componente da classe *Experience*, mas possui a propriedade de ser *independente do contexto* de captura, ou seja, a informação capturada e os hiperdocumentos gerados automaticamente para o contexto de aulas têm a mesma estrutura de dados que hiperdocumentos gerados para o contexto de reuniões, por exemplo, devido à abordagem de dados de primeira classe para captura de sessões (em qualquer contexto, as sessões capturadas podem conter *streams* de áudio e vídeo, slides, *strokes* e seus outros componentes).

Figura 19 – O modelo conceitual para o suporte à autoria evolucionária, definido como uma extensão ao modelo de geração automática de hiperdocumentos.

A informação de contexto é definida na classe *Experience* através de uma propriedade genérica *contextInfo*; essa classe deve ser instanciada para cada contexto que o sistema de autoria queira suportar, e os dados específicos serão criados, de acordo com o contexto, como propriedades *contextInfo* que permitirão o tratamento das especificidades de cada contexto. É a classe *Experience* que representa os dados e metadados de uma experiência capturável da vida real, como uma aula ou uma reunião, e que define as operações que podem ser realizadas sobre esses dados. Uma experiência pode ser composta por zero ou mais sessões; uma experiência que ainda não contenha sessão associada terá sido preparada previamente, e estará agendada para ocorrer em determinada data. As propriedades *contextInfo* podem ser atributos de classe, ou até mesmo instâncias de outras classes do modelo, como *PrepElement* e *ExpElement*, que ainda serão apresentadas.

A classe *Place* define um contêiner lógico para as experiências, indicando a localização lógica das mesmas. Um objeto *place* pode estar contido em outros objetos do mesmo tipo, formando cadeias *n-place* hierárquicas e estruturadas, como, por exemplo, a hierarquia *Turma* $\subset$ *Disciplina* $\subset$ *Curso* no contexto da experiência *Aula*. Um *place* deve definir operações para adicionar e remover tanto experiências como outros *places*.

A classe *Reusable* representa uma generalização de uma unidade de informação de autoria independente, representadas nesse modelo pelas classes *ExpElement* (de Elemento de Expansão) e *PrepElement* (de Elemento de Preparação). Essas classes representam qualquer informação inserida em um sistema de autoria, e devem ser estruturadas de acordo com a propriedade *type* (tipo da informação) e *value* (valor da informação). Um *PrepElement* é uma informação que pode ser reutilizada, e que fica disponível para o autor que deseja criar novas informações a partir desses elementos. Um *ExpElement* é uma unidade de informação que pode ser alterada (expandida) com novas informações. Os objetos *Experience* podem conter diversos objetos *PrepElement* e *ExpElement*, de acordo com o contexto.

Por fim, a classe *Reference* representa a informação de referência entre elementos de uma sessão; cada um dos objetos *Session* pode conter zero ou mais referências, que são compostas por um ponteiro indicando origem e destino da referência, e a data de criação da mesma; também deve oferecer aos autores as operações de listagem e remoção das referências criadas entre as sessões capturadas.

A extensão ao modelo conceitual de Pimentel et al. (2003) apresentada acima foi proposta para oferecer uma abordagem satisfatória aos problemas identificados no projeto dos sistemas de autoria. O Problema do Contexto é contemplado nesse modelo ao definir-se a classe *Experience* de maneira genérica, porém com um componente dependente de contexto, que deve ser implementado quando da instanciação dessa classe no desenvolvimento de um sistema de autoria para a informação multimídia evolucionária. O Problema da Localização é resolvido através da estruturação das experiências em cadeias *n-places* hierárquicas; já o Problema do Reuso é abordado através da definição da classe *Reusable* de elementos de informação, e de suas especializações *PrepElement* e

ExpElement. A definição dessas classes, aliada à definição da classe *Reference* como elemento das sessões, também contempla as deficiências em relação ao suporte à autoria interativa do modelo de geração automática, que é um modelo estático e não prevê a alteração das informações capturadas.

Como o modelo original, o modelo definido também é um modelo aberto, que pode ser utilizado para o desenvolvimento de aplicações de autoria de informação multimídia em qualquer ambiente de captura, independentemente de contexto ou plataforma.

5.5. Considerações Finais

Neste capítulo foram apresentados os principais problemas identificados no projeto de sistemas de autoria de informação multimídia evolucionária, e proposto um modelo aberto de suporte a essa autoria, definido como uma extensão ao modelo conceitual para geração automática de hiperdocumentos em ambiente de captura proposto por Pimentel et al. (2003). A definição desse modelo era um dos objetivos desse trabalho.

A seguir, será apresentado o sistema AutorE, que foi desenvolvido de acordo com o modelo aberto de suporte à autoria proposto nesse capítulo.

6. O sistema AutorE

6.1. Considerações Iniciais

Conforme apresentado no Capítulo 3, os sistemas de captura e acesso de experiências cotidianas que trabalham com informação multimídia evolucionária não oferecem um suporte satisfatório às características de autoria dessa informação. Mesmo os sistemas mais recentes, como o iClass (Cattelan et al., 2002), são focados exclusivamente na geração automática de hiperdocumentos, não contemplando outras modalidades de autoria.

O AutorE (Sistema de Autoria Evolucionária) foi concebido como um sistema aberto, estruturado e baseado nos padrões XML apresentados no Capítulo 4. A disponibilização das funcionalidades necessárias para que as modalidades de autoria em informação multimídia evolucionária (preparação, referência e expansão) sejam suportadas permitem que sistemas de captura utilizem essa estrutura para oferecer uma experiência de autoria baseada nessas modalidades. O seu desenvolvimento foi realizado baseando-se no modelo aberto apresentado no Capítulo 5, o que permitiu implementar as características de estruturação e suporte desejadas para o sistema.

Neste capítulo, serão apresentados o conjunto de requisitos definidos para o AutorE, bem como sua modelagem conceitual e sua arquitetura. Também serão discutidos os aspectos de implementação dos módulos do sistema, e a possibilidade de extensão a outros contextos. Por fim, serão apresentados dois cenários de uso.

6.2. Definição dos Requisitos

Para a definição dos requisitos do sistema AutorE, foram levados em conta um conjunto de funcionalidades que permitissem o suporte à autoria referente aos aspectos da informação multimídia evolucionária nos tempos “Antes da Sessão” e “Após a Sessão” - expansão progressiva e referência intrínseca (ver Seção 2.2) e que também suportasse a Autoria de Preparação, com o reuso de material prévio (ver Seção 3.1). Além disso, alguns

requisitos não-funcionais, como a plataforma do sistema, foram considerados. Os requisitos são os seguintes:

1. Integrar a informação em uma hiperbase aberta. Tanto a informação capturada quanto a informação de autoria devem estar armazenadas em uma base de dados aberta, permitindo que os dados sejam facilmente exportados e acessados por quaisquer aplicações.
2. Relacionar porções de uma sessão capturada com sessões anteriores. Considerando que existam sessões capturadas e armazenadas na base, e o tempo “Após a Sessão”, o sistema deve permitir que:
 - O autor relacione (crie uma ligação entre) uma sessão completa e uma ou mais sessões armazenadas anteriormente;
 - O autor relacione (crie uma ligação entre) uma sessão completa e uma ou mais partes (por exemplo, trecho de áudio, slides) de sessões armazenadas anteriormente;
 - O autor relacione uma parte de uma sessão (por exemplo, trecho do áudio ou um slide) com uma ou mais sessões armazenadas anteriormente;
3. Preparar uma nova sessão a partir de dados utilizados na preparação de outras sessões. Considerando o tempo “Antes da Sessão”, o sistema deve:
 - Armazenar o material original de preparação de sessões (slides, textos, etc.) e metadados sobre os mesmos.
 - Oferecer a possibilidade de utilizar material prévio para a criação de uma nova sessão.
 - Criar uma nova sessão a partir do material previamente armazenado.
 - Tornar disponível qualquer material novo para preparação de futuras sessões.
4. Preparar uma nova sessão a partir de dados capturados em outras sessões. Também no tempo “Antes da Sessão”, o sistema deve:

- Oferecer a possibilidade de utilizar material previamente capturado para a criação de uma nova sessão.
 - Criar a nova sessão a partir do material previamente capturado e armazenado no sistema.
5. Criar novos documentos associados a sessões a partir de documentos já existentes.
Considerando o tempo “Após a Sessão”, o sistema deve:
 - Recuperar um documento (gerado a partir dos dados capturados).
 - Permitir a inserção de novas informações nesse documento (texto, mídias).
 - Armazenar o documento estendido pelas novas informações.
 6. Suportar a autoria nos contextos das experiências Aulas e Reuniões.
 7. Ser construído como uma aplicação Web modular, no contexto do projeto InCA-SERVE, permitindo a expansão de suas funcionalidades com o suporte a novos contextos de experiências, no futuro.
 8. Utilizar a plataforma JavaServer Pages (JSP), conforme discutido na Seção 4.5.2.

O conjunto de requisitos acima foi definido como o escopo do AutorE para esse trabalho, suprimindo as principais necessidades de suporte à autoria que são demandadas nos sistemas InCA-SERVE. No entanto, ele ainda apresenta algumas limitações como, por exemplo, a ausência do suporte à autoria no tempo “Durante a Sessão”, ou seja, permitir que as informações sejam modificadas ao mesmo tempo em que são capturadas durante as experiências de captura.

6.3. Modelagem Conceitual

A modelagem conceitual do AutorE foi realizada como uma instanciação do modelo aberto de suporte à autoria evolucionária, apresentado no Capítulo 5. Foram considerados os dois contextos que deveriam ser suportados inicialmente pelo sistema (Aulas e Reuniões), e criado o modelo com as classes correspondentes. Essa modelagem é apresentada na Figura 20.

Figura 20 – Modelagem conceitual do sistema AutorE.

As classes principais da modelagem, *Aula* e *Reuniao*, são instâncias de *Experiences* para o contexto de aulas e reuniões, respectivamente. As classes *Participante* e *ItemPauta* representam propriedades *contextInfo*, bem como *Ata* e *Tarefa*, que além de informação de contexto, também representam objetos *ExpElement*.

Da mesma forma, as classes *Mídia*, *URL* e *BGSlide* representam objetos *ExpElement*, sendo que este último também é instanciado como um objeto *PrepElement* pois os slides utilizados na preparação de sessões de captura para as experiências correspondentes são disponibilizados para autoria de novas sessões (reuso de material prévio).

As classes *Disciplina* e *Turma* representam *Places* no contexto de aulas, enquanto que a classe *Sala* faz esse mesmo papel no contexto de reuniões. As classes *Sessao* e

Referencia correspondem diretamente às classes *Session* e *Reference* do modelo aberto de suporte à autoria.

Feita a modelagem conceitual, foi estruturada a base de dados que representaria essa modelagem, na forma de coleções de documentos XML, mapeando as classes e propriedades da modelagem. Essa estrutura de coleções, projetada para banco de dados XML nativo, é apresentada na Figura 21.

Figura 21 – Estrutura das coleções de documentos XML representado a base de dados do AutorE.

Exemplos de documentos XML dessas coleções, que contêm os elementos mapeando as demais classes da modelagem, podem ser encontrados no Apêndice A.

6.4. Arquitetura do Sistema

Para o desenvolvimento do AutorE, foi necessária a definição de uma arquitetura para o sistema baseada nas seguintes premissas:

- Simplicidade. Uma arquitetura simples consegue atender a todos os requisitos definidos para o AutorE, e facilita a implantação do mesmo ao exigir menos recursos computacionais.
- Modularidade. Uma arquitetura modular permite que se adicionem novas funcionalidades de forma independente, além de facilitar a manutenção do sistema.
- Acesso Independente. A arquitetura deve definir uma camada de acesso a dados independente de dispositivo, ou seja, definir uma camada abstrata de

acesso que permita a utilização de um novo banco de dados, por exemplo, sem a reprogramação de todo o sistema.

O diagrama geral da arquitetura do sistema AutorE é mostrada na Figura 22. Cada um de seus componentes é apresentado a seguir.

Figura 22 – Diagrama geral de arquitetura do sistema AutorE.

6.4.1. Camada de Acesso

Responsável pelo acesso aos dados armazenados no sistema, é dividida em duas subcamadas: a Camada de Acesso/Dados e a Camada de Acesso/Aplicação. A primeira realiza acesso ao sistema de arquivos do sistema operacional, no tratamento de arquivos de mídia, e ao banco de dados XML nativo (Xindice). A segunda representa a interface entre os módulos de autoria em cada contexto suportado pelo AutorE e os dados armazenados

nos dispositivos representados pela Camada de Acesso/Dados, implementando a abstração de acesso à base.

6.4.2. Módulo de Preparação

Módulo responsável pelo suporte à Autoria de Preparação, ou seja, através dele o usuário poderá realizar as operações de preparação de uma nova sessão para captura, escolhendo o material de outras sessões já armazenadas no sistema e definindo os parâmetros e metadados. No sistema AutorE, foram implementados dois Módulos de Preparação, uma para o contexto de Aulas (AutorE-Class) e outro para o contexto de Reuniões (AutorE-Meet).

6.4.3. Módulo de Referência

Módulo responsável pela criação de referências entre as sessões capturadas, permitindo que se criem *links* entre sessões completas, slides e trechos de áudio/vídeo, internos ou pertencentes a outra sessão, em qualquer combinação dos elementos. Parte da implementação do módulo é compartilhada em ambos os contextos (AutorE-Class e AutorE-Meet) pois a sessão é tratada independentemente de contexto, de acordo com o modelo conceitual apresentado na Seção 6.3.

6.4.4. Módulo de Expansão

O Módulo de Expansão é responsável por implementar as funcionalidades que permitam que se adicionem novas informações às sessões capturadas, e que se altere as informações existentes. Ele suporta a inclusão de texto e anotações em slides, a adição de mídia extras (de qualquer tipo) associadas à sessão, a adição de endereços Web (URLs) para visita e a definição de uma ata, no contexto de Reuniões, ou de tarefas no contexto de Aulas.

6.4.5. Interface Web

A Interface Web é a porção visível do sistema e responsável por integrar os módulos de Preparação, Referência e Expansão através de um *layout* comum a ambos os contextos, para garantir a uniformidade visual. Como qualquer sistema baseado na Web, o AutorE apresenta as mesmas limitações de interface inerentes ao ambiente (Martins Netto, 2002); porém houve a preocupação durante o projeto da interface de oferecer ao usuário um ambiente que permitisse utilizar as funcionalidades do sistema de maneira simples e padronizada.

Na próxima seção, são discutidos os aspectos de implementação referentes aos módulos do AutorE apresentados acima.

6.5. Implementação dos Módulos

Os módulos do AutorE foram implementados de forma a atender as especificações determinadas pelos requisitos e pela arquitetura do sistema. A seguir, será apresentada a forma de implementação dos módulos, que foi a mesma tanto para o AutorE-Class quanto para o AutorE-Meet, e discutidos os detalhes dessa implementação.

6.5.1. Camada de Acesso

A implementação da camada de acesso do AutorE se deu através da definição de classes Java de acesso e manipulação da base (Camada de Acesso/Dados) e criação de componentes JavaBeans que seriam utilizados pelos Módulos de Autoria (Camada de Acesso/Aplicação). A Figura 23 mostra o diagrama de classes da implementação realizada. Essas classes e componentes foram reunidos sob um *package* Java denominado *packDB*.

Figura 23 – Diagrama de classes da implementação referente à Camada de Acesso do AutorE (*packDB*), mostrando os componentes, seus métodos e propriedades.

A Camada de Acesso/Dados é composta pelas seguintes classes e interfaces Java:

- *Class OHDB*: classe estática que realiza o encapsulamento das informações referentes à conexão de dados, como o endereço do servidor de base e o *driver* de conexão.

- *Class ResultSet*: classe que implementa uma abstração de *ResultSets* SQL para bases XML nativas.
- *Interface XMLNativeDB*: define os métodos que devem ser implementados para o suporte a operações em um banco XML nativo em aplicações de autoria. Para que o AutorE acesse qualquer sistema de banco de dados XML, é necessário criar uma classe que implemente essa interface, como a classe *Xindice* descrita abaixo. O código dessa interface é mostrado no Apêndice B.
- *Class Xindice*: Implementação da interface *XMLNativeDB* para acesso ao banco de dados Xindice, que se utiliza da API *XML:DB* para o acesso ao servidor. Implementa, por exemplo, funcionalidades básicas como *queries* XPath e *update* baseado no XUpdate, que são as linguagens suportadas pelo Xindice e definidas como métodos abstratos na interface *XMLNativeDB*.
- *Class DataAccess*: Classe derivada de *Xindice*, determinando qual o banco de dados que será utilizado pelo sistema. Todos os componentes JavaBeans da Camada de Acesso/Aplicação instanciam um objeto *DataAccess* para conexão com a base. Portanto, para substituir a base de dados XML do AutorE, basta criar uma nova classe que implemente a interface *XMLNativeDB* com os métodos adequados para a nova base de dados, e definir a classe *DataAccess* como derivada dessa nova classe criada.

Já a Camada de Acesso/Aplicação é composta pelas seguintes classes (JavaBeans) e interfaces, que manipulam os objetos de autoria definidos no modelo conceitual (ver Seção 6.3):

- *Interface Captureable*: define os métodos que devem ser implementados e as propriedades necessárias para o suporte às operações de autoria em objetos do tipo Experience (ver Seção 5.4) de maneira genérica. Para que o AutorE suporte a autoria em um determinado contexto, é necessário criar uma classe representando esse contexto que implemente essa interface, como as classes *AulaBean* e *MeetBean* descritas abaixo. O código dessa interface é mostrado no Apêndice B.

- *Class AulaBean*: classe que implementa a interface *Captureable* para o suporte à manipulação dos documentos XML no contexto de aulas. Utilizada pelos módulos do AutorE-Class, adiciona propriedades e métodos exclusivos do contexto, como, por exemplo, informações sobre turmas e cursos.
- *Class MeetBean*: classe equivalente à classe *AulaBean*, porém no contexto de Reuniões. Implementa a interface *Captureable*, é utilizada pelos módulos do AutorE-Meet e também adiciona propriedades e métodos exclusivos (por exemplo, as propriedades representando a pauta e os participantes da reunião).
- *Class DisciplinaBean*: implementa o suporte à recuperação de dados referentes a disciplinas, no contexto de aulas.
- *Class TurmaBean*: implementa o suporte à recuperação de dados e à manipulação dos documentos que representam turmas no contexto de aulas (como por exemplo, criar uma nova turma ou adicionar uma aula a uma turma já existente).
- *Class RoomBean*: da mesma forma que a classe *TurmaBean*, essa classe implementa suporte à recuperação de dados e à manipulação de documentos que representam salas, no contexto de reuniões.
- *Class SessionBean*: essa classe implementa o suporte às operações de autoria que são realizadas sobre as sessões capturadas, independentemente de contexto, pois a base de sessões capturadas é comum.

Todos os componentes JavaBeans instanciam um objeto *conn* da classe *DataAccess*, e as operações de inserção e atualização dos documentos na base de dados são realizados através de chamadas a métodos desse objeto. Para o tratamento dos arquivos XML referentes a essas operações, são utilizados os métodos do *parser* DOM disponível na API JAXP.

Para a manipulação das operações de *upload* e armazenamento de arquivos de mídia no sistema de arquivos, foi utilizado um componente gratuito para utilização não-comercial

(*UploadBean*), desenvolvido pela JavaZoom e que já implementava todas as operações necessárias para o AutorE. Esse componente pode ser obtido em <http://www.javazoom.net/jzservlets/uploadbean/uploadbean.html>.

6.5.2. Módulos de Autoria

A implementação de todos os módulos de autoria (Preparação, Referência e Expansão), em ambos os contextos, foi realizada de forma semelhante. Os módulos foram desenvolvidos como aplicações Web na plataforma JSP, compostas por páginas dinâmicas, formulários e *scripts* JSP, que recebem dados e manipulam os documentos XML na base através de chamadas a componentes da Camada de Acesso, descrita na seção anterior.

A estrutura básica de funcionamento dos módulos é mostrada na Figura 24. Para realizar uma operação, o usuário utiliza formulários ou páginas dinâmicas, que recuperam informações da base XML e oferecem a possibilidade de que essas informações sejam alteradas de alguma forma, de acordo com a operação e o módulo. Essas novas informações são enviadas para um *script* de processamento, que faz a validação das mesmas, executa as alterações nos documentos XML armazenados na base e retorna o status da operação, o que possibilita ao usuário de repetir a operação ou realizar quaisquer outras operações oferecidas por qualquer outro módulo através do mesmo processo.

Figura 24 – Esquema ilustrando a estrutura de funcionamento dos módulos do AutorE.

Os *scripts* JSP e páginas dinâmicas utilizam os componentes JavaBeans disponibilizados pela Camada de Acesso/Aplicação para a recuperação e processamento dos dados, de acordo com o contexto e a operação realizada; a vantagem dessa abordagem é a facilidade de reuso de componentes e de alteração de funcionalidades, pois na maioria das vezes, basta reprogramar as operações no componente, sem a necessidade de se alterar os *scripts* do sistema. Um exemplo de um *script* que recebe informações de um formulário, e adiciona uma nova URL a uma reunião (no Módulo de Expansão do AutorE-Meet) através de uma chamada a uma operação do componente *MeetBean* é mostrado na Figura 25.

```

01 <%@page import="java.util.*" %>
02
03 <jsp:useBean id="Meet" scope="page" class="packDB.MeetBean"/>
04
05 <%@ include file="config/config.jsp" %>
06
07 <%
08 String roomid = new String();
09 String meetid = new String();
10
11
12 String href = request.getParameter("href");
13 String desc = request.getParameter("desc");
14 roomid = request.getParameter("roomid");
15 meetid = request.getParameter("meetid");
16
17 // save XML data
18 String status = Meet.addURL(meetid, href, desc);
19
20 %>
21
22 <jsp:forward page="expmeeturl.jsp">
23 <jsp:param name="room" value="<%=roomid%>" />
24 <jsp:param name="id" value="<%=meetid%>" />
25 </jsp:forward>

```

Figura 25– Script de processamento no módulo de expansão do AutorE-Meet. O componente *MeetBean* é instanciado (linha 3); as informações provenientes do formulário (linhas 12 a 15) são recuperadas e é executada a alteração no documento correspondente, na base de dados, através da chamada ao método *addURL* do componente *MeetBean* (linha 18).

A seguir, serão discutidos os aspectos específicos relacionados às funcionalidades implementadas de cada um dos módulos de autoria, incluindo a apresentação da interface Web desenvolvida para eles.

6.5.3. Módulo de Preparação

Para o Módulo de Preparação, foi implementada a funcionalidade de Criar Nova Aula (AutorE-Class) e Criar Nova Reunião (AutorE-Meet), dando ao usuário a possibilidade de inserir material novo, e de reutilizar ou não materiais previamente

capturados ou utilizados para preparação de outras aulas e reuniões. A Figura 26 mostra a interface de criação de uma nova aula, no Módulo de Preparação do AutorE-Class. Alguns dados do formulário são dinâmicos, provenientes da base de dados XML, como, por exemplo, o *select box* para escolha do Curso.

The screenshot shows a web browser window titled 'AutorE - Microsoft Internet Explorer' with the address 'http://localhost:8080/autore/newclass.jsp?reuse=1'. The page features a navigation bar with buttons: 'Preparação', 'Referência', 'Expansão', and 'Sair'. The main content area is titled 'Módulo de Preparação' and contains a form for creating a new lesson ('Nova Aula').

The form fields are as follows:

- Curso:** A dropdown menu with 'Hipermidia (SCE-5811)' selected.
- Período:** Two dropdown menus with '1o' and '2003' selected.
- Título:** A text input field containing 'Paradigmas'.
- Data:** A text input field containing '12/05/2003'.
- Assunto:** A text input field containing 'novos paradigmas de hipertexto'.
- Resumo:** A text input field containing 'apresentação dos paradigmas'.

Below the form, there are three checkboxes, all of which are checked:

- ☒ Quero inserir slides novos. Arquivo de slides: C:\Meus documentos\se [Procurar...]
- ☒ Quero aproveitar slides usados na preparação de aulas anteriores.
- ☒ Quero aproveitar slides capturados em aulas anteriores.

At the bottom of the form, there are two buttons: 'Voltar' and 'Avançar >>'.

Figura 26 – Formulário de criação de uma nova aula, no AutorE-Class.

O arquivo com novos slides a ser inserido deve ser um arquivo ZIP especial, que contenha as imagens dos slides (em qualquer formato) e um arquivo XML descrevendo os metadados de cada slide. Esse arquivo ZIP é então enviado, processado e descompactado por uma classe criada para esse fim (classe *ZipUtil*), e os arquivos de slides são armazenados no servidor.

Para o reaproveitamento de material anterior, é utilizado o formulário exemplificado na Figura 27, que permite a escolha dos slides que se deseja reaproveitar. O sistema apresenta todos os slides disponíveis referentes somente ao mesmo Curso (ou Sala, no

AutorE-Meet) da aula ou reunião que está sendo criada. Essa é uma limitação de funcionalidade que foi definida devido à ausência de integração, no momento, com um mecanismo de busca, o que dificultaria a seleção devido ao grande volume de dados. Ao clicar-se no número do slide, é aberta uma janela para que se possa visualizá-lo.

Módulo de Preparação

Nova Aula - Escolher Slides

Curso: SCE-185

Slides de Preparação disponíveis:

- 2o Semestre - 2002
 - Apresentacao (12/12/2003) ☐ 0 ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ☐ 7 ☐ 8 ☐ 9
 - Aula de Exercicios (12/12/2003) ☐ 0 ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6
 - Aula Extra (13/12/2002) ☐ 0 ☐ 1 ☐ 2 ☒ 3 ☐ 4 ☐ 5 ☒ 6 ☐ 7 ☐ 8 ☐ 9 ☐ 10 ☐ 11 ☐ 12
 - Teoria das Filas (12/12/2002) ☐ 0 ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☒ 6 ☐ 7 ☐ 8 ☐ 9 ☐ 10 ☐ 11 ☐ 12
- 1o Semestre - 2002
 - Teoria dos Grafos (12/06/2002) ☐ 0 ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☒ 6 ☒ 7 ☒ 8 ☐ 9 ☐ 10 ☐ 11 ☐ 12

Curso: SCE-185

Slides Capturados disponíveis:

- 2o Semestre - 2002
 - Aula de Exercicios (12/12/2003) Sessão 1: ☐ 0 ☐ 1 ☐ 2 ☐ 3 ☒ 4 ☐ 5 ☐ 6 ☐ 7 ☐ 8 ☐ 9 ☐ 10 ☐ 11 ☐ 12
 - Aula Extra (13/12/2002) Sessão 1: ☐ 0 ☐ 1 ☐ 2 ☐ 3 ☒ 4 ☐ 5 ☒ 6 ☒ 7 ☐ 8 ☒ 9 ☐ 10 ☐ 11 ☐ 12

Voltar **Avançar >>**

Figura 27 – Formulário para escolha dos slides a serem reaproveitados, ao criar-se uma nova aula no AutorE-Class.

Ao término do processo, existe ainda um formulário onde o usuário define a ordem de apresentação dos slides, incluindo os slides novos e reaproveitados. O sistema então cria um documento XML correspondente à aula ou reunião baseado nas informações do usuário, e atualiza os documentos de Curso/Turma ou Sala correspondentes. No AutorE-Class, um novo documento de Turma é criado automaticamente caso não exista a turma indicada no ato de criação de uma nova aula. Exemplos dos documentos XML criados e manipulados pelos módulos do AutorE podem ser encontrados no Apêndice A.

6.5.4. Módulo de Referência

Este módulo implementa a funcionalidade de criação de referências entre sessões. Um formulário inicial pede que se escolha o Curso/Turma (no AutorE-Class) ou a Sala (AutorE-Meet) para a filtragem das sessões disponíveis, devido à mesma limitação de busca descrita na seção anterior. Escolhido esse filtro, são apresentadas as sessões capturadas em um outro formulário, para que o usuário defina a sessão de origem e a sessão destino da referência.

Escolhidas as sessões de origem e destino, é apresentado o formulário mostrado na janela de fundo da Figura 28. O sistema permite que se criem referências entre uma sessão completa, um slide ou um trecho de áudio e vídeo. Para selecionar um slide, basta clicar no número do slide, que uma nova janela se abre para visualização do mesmo e seleciona-o automaticamente, ou então digitar o número do slide. Para selecionar um trecho de áudio e vídeo, também é possível digitar o tempo de início (em segundos) na caixa de entrada, porém clicando-se no *link* “[\(veja/ouça e escolha o trecho\)](#)” é aberta a janela mostrada na parte inferior direita da Figura 28, onde um *applet* executa o áudio e vídeo, atualizando automaticamente o tempo corrente de reprodução na caixa de texto logo abaixo. Clicando-se no botão “Selecionar”, esse tempo é transferido para o formulário principal.

A automatização de formulários foi implementada através de funções JavaScript. O *applet* que faz a reprodução do áudio/vídeo é encontrado como um exemplo no pacote *Java Media Framework*, e foi alterado para que disponibilizasse métodos de sincronização que pudessem ser lidos pelas funções de automatização do formulário.

Figura 28 – Interface do Módulo de Referência, usando como exemplo o AutorE-Meet. Na janela maior ao fundo, o formulário de criação de referências entre sessões; no canto inferior direito, a janela de escolha do trecho de áudio/vídeo da sessão, onde o *applet* está reproduzindo o vídeo capturado; no canto superior direito, a janela com a listagem das referências de uma sessão.

É possível visualizar as referências já criadas clicando-se sobre o título da sessão desejada, de origem ou destino. A janela mostrada no canto superior direito da Figura 28 é apresentada, listando as referências que têm a sessão escolhida como origem, e podem ser removidas através de um clique no ícone em forma de “X” que vem antes da descrição de cada referência.

O Módulo de Referência altera os documentos XML na base de Sessões capturadas para que eles passem a incluir informações das referências criadas entre as mesmas; portanto, esse módulo não cria novos documentos representando sessões; esses documentos são criados pelas aplicações de captura, como o iClass.

6.5.5. Módulo de Expansão

No Módulo de Expansão são implementadas as funções de expansão das informações capturadas nas aulas e reuniões, divididas em quatro áreas: slides, mídias, URLs e ata (no caso de reuniões) ou tarefas (no caso de aulas). Da mesma forma que no módulo de Referência, primeiramente é necessário escolher o filtro por Curso/Turma ou Sala para listagem das aulas e reuniões disponíveis, e apresentação das opções para expansão, através de um menu como o apresentado na Figura 29.

Figura 29 – Menu dinâmico listando aulas disponíveis em uma turma, que oferece ao usuário as opções de expansão no AutorE-Class.

A opção de Slide apresenta um novo menu dinâmico listando todos os slides disponíveis para edição; ao clicar em um slide, é apresentado um formulário onde se pode editar ou adicionar o texto correspondente ao slide, e também anotações que se queira

acrescentar. Esta opção faz com que sejam atualizados os arquivos XML correspondentes na base de Sessões do sistema, que contém as informações sobre os slides capturados.

A opção Tasks (Ata no AutorE-Meet) permite que se adicione tarefas após o término da aula ou se defina uma ata para uma reunião, através de um formulário contendo um campo de texto destinado à edição dessa informação, que é alterada no documento XML correspondente à Aula ou à Reunião escolhida.

As opções Mídia e URL são idênticas nos dois contextos. Elas permitem que se adicionem ou removam mídias extras (de qualquer natureza) e URLs, formando um catálogo de endereços cuja visita é recomendada, dentro do assunto da aula ou reunião. A interface implementada nessas duas opções é bastante similar, e mostrada na Figura 30. Um formulário apresenta em seu lado esquerdo o espaço para *upload* de um arquivo de mídia, e uma descrição do mesmo. No seu lado direito, são mostrados os arquivos já inseridos na aula/reunião escolhida, oferecendo a possibilidade de removê-los através de um clique no ícone em forma de “X”, ou de visualizá-los, clicando-se sobre a descrição dos mesmos. A interface para adição de URLs difere apenas no fato de que o espaço para *upload* de arquivo é substituído por uma caixa de texto para digitação do endereço Web desejado. Ambas as opções causam alteração no documento XML correspondente que é armazenado na base de Aulas ou Reuniões.

A interface de todos os módulos de autoria foi implementada baseando-se em XHTML, tanto no código estático das páginas quanto no conteúdo dinâmico gerado por JSP. Todas as páginas dos módulos incluem arquivos CSS para definição de *layout*, e arquivos dinâmicos de cabeçalho e rodapés que servem tanto para inicialização do ambiente JSP (variáveis de sessão e arquivos de configuração) quanto para a uniformização do layout e identificação de contexto, pois esses arquivos são únicos e são utilizados tanto pelo AutorE-Class quanto pelo AutorE-Meet.

Na próxima seção serão discutidos os aspectos relacionados à extensão das funcionalidades do AutorE a outros contextos.

Figura 30 – Formulário de inserção de mídias no Módulo de Expansão do AutorE-Meet. À direita, a listagem das mídias já adicionadas a esta reunião.

6.6. Utilização em outros contextos

Como já apresentado, neste trabalho foi implementado o suporte à autoria evolucionária em dois contextos distintos: o de Aulas, implementado pelo AutorE-Class, e o de Reuniões, implementado pelo AutorE-Meet. Porém, o sistema AutorE pode ser estendido para suportar outros contextos, devido a sua estrutura modular e à existência de uma camada de acesso bem definida em sua arquitetura.

A implementação do suporte a um novo contexto no AutorE envolve os seguintes passos, supondo que a modelagem conceitual para o novo contexto, baseada no modelo apresentado no Capítulo 5, seja efetuada:

1. Criação de um componente JavaBean que implemente a interface *Captureable* e defina os métodos e propriedades específicas do contexto;
2. Criação dos componentes *Places* (contêineres) necessários à estruturação da informação das experiências capturadas.
3. Implementação das páginas, formulários e *scripts* dos módulos de autoria. Para isso, pode-se utilizar como base os módulos já existentes; existe uma padronização dos nomes das páginas que os compõem, diferenciando apenas o nome do contexto (por exemplo, a página do formulário de inserção de uma referência chama-se *refclass.jsp* no AutorE-Class, e *refmeet.jsp* no AutorE-Meet).
4. Alteração das páginas comuns do sistema para que suportem o novo contexto. São poucas páginas desse tipo, e, portanto, a alteração no sistema será mínima. A Figura 31 mostra a página de seleção de contexto do sistema, que é uma dessas páginas que demandariam de alteração.

Figura 31 – A página de seleção de contexto do sistema AutorE.

Essa abordagem é trabalhosa, mas não é complexa, pois os módulos são independentes e a maior parte do trabalho de implementação de um novo contexto consiste

na criação de novas classes Java e páginas JSP, baseada nas classes e páginas já existentes. A estrutura dos arquivos componentes do sistema, incluindo a listagem das páginas comuns que devem ser alteradas no caso da implementação de um novo contexto, pode ser encontrada no Apêndice C.

6.7. Cenário de Uso

O cenário de uso típico para o AutorE são ambientes onde existam aplicações de captura de informação multimídia evolucionária e geração automática de hiperdocumentos, porém que não suportam as características de autoria evolucionárias dessas informações, como por exemplo, os ambientes iClass e eMeet, descritos no Capítulo 2.

No ambiente iClass, por exemplo, o professor poderia utilizar o AutorE-Class para criar uma nova aula de Hipermissão, utilizando para isso alguns slides que já tenham sido capturados no semestre anterior, adicionando também novos slides. Após a aula, seria possível montar uma lista de sites para que os alunos, quando estivessem assistindo àquela aula capturada, navegassem e obtivessem mais informações. Também seria possível relacionar a aula dada a outra aula do semestre anterior, criando uma referência de forma a complementar o entendimento da aula.

No eMeet, da mesma forma, uma reunião para revisão de projeto poderia ser criada através do AutorE-Meet, a partir da reunião onde foi decidida a especificação; ao término da reunião, o secretário poderia acrescentar a ata, que estaria disponível em um posterior recuperação dos dados, para a visualização dessa mesma reunião. Também poderia acrescentar o arquivo da apresentação multimídia do produto em questão para que os participantes pudessem vê-lo com mais tempo em casa, pelo computador.

6.8. Considerações Finais

Neste capítulo, foi apresentado o sistema AutorE, implementado com base no modelo aberto de suporte à autoria de informação multimídia evolucionária proposto no Capítulo 5 e que suporta a autoria em dois contextos: o contexto de Aulas (AutorE-Class) e o contexto de Reuniões (AutorE-Meet).

O AutorE apresenta como vantagem o fato de ser um sistema aberto baseado em padrões XML, definindo uma base estruturada e uma arquitetura modular que permitem a extensão de suas funcionalidades e a integração com outros sistemas. No entanto, o sistema apresenta ainda algumas limitações, como a falta de integração com um mecanismo de busca em sessões e a ausência de suporte à autoria no tempo “Durante a Sessão”.

No próximo capítulo, serão apresentadas as conclusões, contribuições e trabalhos futuros referentes ao trabalho desenvolvido.

7. Conclusões

Como resultado da dificuldade em assimilar e manter a atenção a todas as informações a que as pessoas estão submetidas em seu cotidiano, foram desenvolvidos vários sistemas baseados na captura dessas informações, principalmente nos contextos educacional tradicional e de suporte a reuniões. A maioria desses sistemas, no entanto, não suporta operações de autoria interativa sobre essa informação capturada, o que compromete a utilidade dos mesmos.

Este trabalho teve como objetivo investigar um *modelo aberto* para o suporte às características de autoria da informação capturada, e desenvolver um *sistema aberto* baseado neste modelo, que implementasse as funcionalidades de autoria e que pudesse ser integrado a outros sistemas.

7.2. Contribuições

As contribuições desse trabalho são:

1. Identificação, estudos e comparação entre sistemas de captura de informações em relação às suas características de autoria (Capítulo 3).
2. Definição de um modelo aberto de suporte à informação multimídia evolucionária como extensão ao modelo de geração automática de hiperdocumentos do projeto InCA-SERVE (Capítulo 5).
3. Implementação de uma camada de abstração de dados para acesso, recuperação e modificação de documentos em bancos de dados XML nativos (Seção 6.5.1).
4. Definição de uma interface Java que representa objetos do tipo *Experience*, que são as experiências capturáveis em um contexto específico (Seção 6.5.1).
5. Definição de um sistema Web aberto, modular e expansível, baseado no modelo de autoria proposto, para o suporte às operações de autoria em informação multimídia evolucionária, denominado AutorE (Seções 6.2, 6.3 e 6.4).

6. Implementação dos módulos do AutorE responsáveis pelo suporte à autoria no contexto educacional (AutorE-Class) (Seção 6.5).
7. Implementação dos módulos do AutorE responsáveis pelo suporte à autoria no contexto de reuniões (AutorE-Meet) (Seção 6.5).

7.3. Trabalhos Futuros

Como trabalhos futuros que possam ser derivados do trabalho apresentado, temos:

1. Integração de um mecanismo de busca na base aberta de hiperdocumentos, para localização avançada de aulas e reuniões em diversas salas e turmas, ampliando a as funcionalidades de referência e expansão do AutorE.
2. Integração de um mecanismo de autenticação e autorização de usuários no sistema, para controle de acesso aos recursos do mesmo. Não foi desenvolvido um novo sistema deste tipo para o AutorE, devido à grande quantidade de sistemas gratuitos disponíveis, como por exemplo, autenticação via LDAP¹⁹. O AutorE já está preparado para essa integração a um sistema de autenticação, no entanto, isso ainda não foi feito.
3. Integração com o sistema iClass, para que ele possa incorporar as funcionalidades de autoria oferecidas pelo AutorE no contexto de aulas. Como os dois sistemas foram projetados em uma mesma plataforma aberta, essa integração requer pouco esforço.
4. Integração com o sistema eMeet para que ele possa incorporar as funcionalidades de autoria oferecidas pelo AutorE no contexto de reuniões. Como o eMeet foi construído sobre uma plataforma proprietária, seria necessário um esforço maior para que essa integração fosse feita.
5. Integração com o sistema CoTeia, o que permitiria adicionar funcionalidades de compartilhamento e autoria colaborativa ao sistema AutorE.

¹⁹ LDAP (*Lightweight Directory Access Protocol*) é um protocolo de autenticação e permissão de usuários baseado em diretórios. Existem várias implementações gratuitas do protocolo LDAP, e a mais conhecida delas é o OpenLDAP (<http://www.openldap.org>).

6. Estender as funcionalidades do sistema AutorE para utilização durante a captura de uma sessão, ou seja, suportar a autoria no tempo “Durante a Sessão”, permitindo que os dados preparados sejam modificados à medida em que são capturados. Foi implementado apenas o suporte à autoria nos tempos “Antes” e “Depois da Sessão”.
7. Implementação de funcionalidades de cadastro-base das informações de Salas e Disciplinas no AutorE. Atualmente, não há interface para a criação dessas informações, sendo realizado a inserção diretamente pela base de dados.

7.4. Considerações Finais

Com a realização deste trabalho, o objetivo de prover um modelo aberto para o suporte à autoria de informação multimídia evolucionária e implementar um sistema que oferecesse esse suporte foi atingido. O sistema AutorE foi projetado como um sistema aberto, baseado no modelo investigado, e pode ser integrado a outros sistemas no contexto InCA-SERVE, como o iClass. Ele ainda apresenta algumas limitações, que podem ser eliminadas através de trabalhos futuros colaborando para a evolução do sistema, mas já oferece um suporte amplo aos aspectos de autoria mais importantes da informação multimídia evolucionária.

Referências

- (Abowd, 1999) Abowd, G. Classroom 2000: an experiment with the instrumentation of a living educational environment. *IBM Systems Journal*, vol. 38, n. 4, p. 508-530, Out/1999.
- (Abowd et al., 1998a) Abowd, G.; Atkenson, C; Brotherton, J.; Enqvist, T.; Gully, P.; Lemon, J. Investigating the capture, integration and access problem of ubiquitous computing in an educational setting. *Proceedings of CHI'98, ACM, New York* p. 440-447, 1998.
- (Abowd et al., 1998b) Abowd, G.; Brotherton, J.; Bhalodia, J. Classroom 2000: a system for Capturing and Accessing Multimedia Classroom Experiences. *Proceedings of CHI'98, ACM, Mai/1998*.
- (Arruda Jr & Pimentel, 2001) Arruda Jr., C.R.E.; Pimentel, M.G.C. Projeto e Implementação de um Sistema Colaborativo de Edição. *Revista Eletrônica de Iniciação Científica da Sociedade Brasileira de Computação (REIC-SBC)*, Ano 1. Nov/2001.
- (Baldochi Jr., 2002) Baldochi Jr., L.A. Armazenamento, Recuperação e Extensão de Informação Proveniente de Atividades de Captura. *Qualificação de Doutorado, ICMC-USP*, Mar/2002.
- (Bila, 2002) Bila, W.S. eMeet/InCA: Captura e Acesso de Informação Multimídia em Reuniões Distribuídas. *Dissertação de Mestrado, ICMC-USP*, Mar/2002.
- (Bourret, 2003) Bourret, R. XML and Databases.
<http://www.rpbourret.com/xml/XMLAndDatabases.htm>, Jan/2003.
- (Brotherton et al., 1998) Brotherton, J.A.; Abowd, G.D. Rooms Take Notes! *Working Papers of Aai'98 Spring Symposium*, Mar/1998

- (Cattelan, 2002) Cattelan, R.G. xINCA: construção de aplicações de captura e acesso baseada em montagem de componentes de software. *Plano de Mestrado submetido à FAPESP, ICMC-USP, Maio/2002.*
- (Cattelan et al., 2002) Cattelan, R.G.; Andrade, A.R.; Penedo, C.F.R.; Pimentel, M.G.P. iClass: um sistema para captura e acesso de sessões em ambiente educacional, *REIC - Revista eletrônica de Iniciação Científica*, 2002.
- (Conklin, 1987) Conklin, J. Hypertext: an introduction and survey. *IEEE Computer*, p.17-41, Set/1987.
- (Dey et al., 1999) Dey, A.K.; Salber, D.; Abowd, G.; Futakawa, M. The Conference Assistant: Combining Context-Awareness with Wearable Computing. *Proceedings of the 3rd International Symposium on Wearable Computers (ISWC '99)*, San Francisco, CA, Out/1999. p. 21-28.
- (Eleutério & Pimentel, 2001) Eleutério, P.M.S.; Pimentel, M.G.P. Geração Sob Demanda de Hiperdocumentos Multimídia a Partir de Informação Capturada em Reuniões Distribuídas. *Anais do I Workshop de Teses e Dissertações do SBMídia`2001*, Out/2001
- (Eleutério, 2002) Eleutério, P.M.S.; eMeet/SERVE: Geração Sob Demanda de Hiperdocumentos Multimídia a Partir de Informação Capturada em Reuniões Distribuídas. *Dissertação de Mestrado, ICMC-USP, Abr/2002.*
- (Ellis et al., 1991) Ellis, C.A.; Gibbs, S.J.; Rein, G.L. Groupware: some issues and experiences. *Communications of the ACM*, 34(1), 38-58, 1991.
- (Guzdial, 1999) Guzdial, M. Collaborative website supporting open authoring. *Submitted to Journal of the Learning Sciences*, 1999
- (Jancke et al., 2000) Jancke, G.; Grudin, J.; Gupta, A. Presenting to local and remote audiences: design and use of the TELEP system. *CHI Letters*, vol. 2, issue 1, p. 384-391, Abr/2000.

- (Macedo et al., 2001) Macedo, A.A.; Bulcão Neto, R.F.; Pimentel, M.G.C. Autoria Colaborativa na Web: experiências e reflexões sobre a CoWeb. *Revista Brasileira de Informática na Educação (RBIE)*, Set/2001.
- (Martins Netto, 2002) Martins Netto, O.A. Heurísticas e Guidelines para Apresentação de Hiperdocumentos Multimídia na Web. *Dissertação de Mestrado, ICMC-USP*, Fev/2002.
- (Minneman et al., 1995) Minneman, S.; Harrison, S.; Janseen, B.; Kurtenback, G.; Moran, T.; Smith, I.; Van Melle, B. A Confederation of Tools for Capturing and Accessing Collaborative Activity. *ACM Conference on Multimedia (Multimedia'95) San Francisco, CA* p. 523-533, Nov/1995.
- (Müeller & Ottmann, 2000) Mueller, R.; Ottmann, T. The “Authoring on the Fly” system for automated recording and replay of (tele) presentations. *ACM/Springer Multimedia Systems Journal*, n. 8(3), p. 158-176, Mai/2000.
- (Mukhopadhyay & Smith, 1999) Mukhopadhyay, S.; Smith, B. Passive Capture and Structuring of Lectures. *ACM Conference on Multimedia (Multimedia'99) Orlando, FL*, Out/1999.
- (Myers, 2001) Myers, B. Using Handhelds and PCs Together. *Communication of the ACM*, vol 4, n. 11, Nov/2001.
- (Nielsen, 1992) Nielsen, J. Finding Usability Problems Through Heuristic Evaluation. *Proc. of CHI'92, ACM, Monterey, CA*, p.373-380, 1992.
- (Nielsen, 1993) Nielsen, J. Usability Engineering. *Academic Press, London, United Kingdom*, 1993.
- (Perens, 2003) Perens, B. Open Standards: Principles and Practice. <http://perens.com/OpenStandards/Definition.html>, 2003.

- (Pimentel et al., 1999) Pimentel, M.G.C.; Abowd, G. Development and Understanding of Automated Capture Environments to Support Long-Term Use. *Projeto de Cooperação Internacional PROTEM-CC-CnPQ/Brasil e NSF/EUA*, Out/1999
- (Pimentel et al., 2000) Pimentel, M.G.C.; Abowd, G. Ishiguro, I. Linking by Interacting: a Paradigm for Authoring Hypertext. *Proc. ACM Hypertext'00, San Antonio, TX*, Mai/2000.
- (Pimentel & Teixeira, 2000) Pimentel, M.G.C.; Teixeira, C.A.C. XML: Explorando suas aplicações na Web. *JAI 2000-SBC, XIX Jornada de Atualização em Informática*, 2000.
- (Pimentel et al., 2001) Pimentel, M.G.C.; Abowd, G.; Kerimbaev, B.; Ishiguro, Y.; Guzdial, M. Supporting Long-Term Educational Activities through Dynamic Web Interfaces. *Journal Interacting with Computers*, Fev/2001.
- (Pimentel et al., 2003) Pimentel, M.G.C.; Izeke, C.A.; Bulcão Neto, R.F.; Fortes, R.P. From Capture to Hypertext: the role of interaction trails and captured media. *Submetido ao ACM Hypertext 2003*.
- (Richter et al., 2001) Richter, H.; Abowd, G.; Geyer, W.; Fuchs, L.; Daijavad, S.; Poltrock, S. Integrating Meeting Capture within a Collaborative Team Environment. *Proceedings of Ubicomp 2001, Atlanta, Georgia, Set/2001*, p. 123-138.
- (Streitz et al., 1994) Streitz, N.A.T.; Geibler, J.; Haake, J.M. E Hol, J. (1994) Dolphin: Integrated Meeting Support Across Local And Remote Desktop Environments And Liveboards. *Proceedings Of Cscw'94*, p. 345-358.
- (Truong & Abowd, 2001) Truong, K.; Abowd, G. Infrastructure for Capture & Access Applications. *Submitted for review to CHI 2002*.
- (Wactlar et al., 1996) Wactlar, H.D.; Kanade, T.; Smith, M.A.; Stevens, S.M. Intelligent Access to Digital Video: Informedia Project, *IEEE Computer*, vol 29, n.o 5, p. 46-52, Mai/1996.

(Weiser, 1993) Weiser, M. Some computer science issues in ubiquitous computing.
Communications of the ACM, vol. 6, n. 7, p. 75-84, Jul/1993.

(White et al., 1998) White, S.A.; Gupta, A.; Grudin, J.; Chesley, H.; Kimberly, G. E
Sanocki, E. A Software System For Education At A Distance: Case Study Results.
Microsoft Research Technical Report Msr-Tr-98-61, Redmond, WA, Nov/1998.

Apêndice A - Exemplos de documentos XML manipulados

Coleção: db/class/aula

```
<?xml version="1.0"?>
<aula id="35">
  <titulo>Seminarios</titulo>
  <autor>dgsante</autor>
  <data>10/04/2003</data>
  <assunto>Seminários dos alunos</assunto>
  <resumo>Temas diversos em computação</resumo>
  <bg_slides textfile="http://localhost:8080/autore/medias/1o_2003/SCE-
5843/35/background/Slide_Texto.xml">
    <bg_slide href="http://localhost:8080/autore/medias/1o_2003/SCE-
5843/35/background/Slide1.JPG" id="0" />
    <bg_slide href="http://localhost:8080/autore/medias/1o_2003/SCE-
5843/35/background/Slide2.JPG" id="1" />
    <bg_slide href="http://localhost:8080/autore/medias/1o_2003/SCE-
5843/35/background/Slide3.JPG" id="2" />
    <bg_slide href="http://localhost:8080/autore/medias/1o_2003/SCE-
5843/35/background/Slide4.JPG" id="3" />
    <bg_slide href="http://localhost:8080/autore/medias/1o_2003/SCE-
5843/35/background/Slide5.JPG" id="4" />
    <bg_slide href="http://localhost:8080/autore/medias/1o_2003/SCE-
5843/35/background/Slide6.JPG" id="5" />
    <bg_slide href="http://localhost:8080/autore/medias/1o_2003/SCE-
5843/35/background/Slide7.JPG" id="6" />
  </bg_slides>
  <secoes />
  <resolucoes />
  <midias />
  <urls />
</aula>
```

Coleção: db/class/disciplina

```
<?xml version="1.0"?>
<disciplina id="SCE-185" nome="Seminários em Computação">
  <creditos />
  <carga_horaria />
  <docentes>
    <docente />
  </docentes>
  <ementa>
    <item_ementa />
  </ementa>
  <bibliografia>
    <item_bibliografia />
  </bibliografia>
</disciplina>
```

Coleção: db/class/turma

```
<?xml version="1.0"?>
<turma disciplina="SCE-185" id="3">
  <periodo>
    <semestre>1</semestre>
    <ano>2003</ano>
  </periodo>
</turma>
```

```

</periodo>
<horario />
<local />
<material_apoio />
<docentes />
<alunos />
<aulas>
  <aula id="20" />
  <aula id="21" />
  <aula id="22" />
  <aula id="23" />
  <aula id="24" />
  <aula id="25" />
  <aula id="26" />
  <aula id="27" />
  <aula id="28" />
  <aula id="29" />
</aulas>
</turma>

```

Coleção: db/meet/reuniao

```

<?xml version="1.0"?>
<reuniao id="19">
  <titulo>Requisitos</titulo>
  <autor>dgsante</autor>
  <resumo>Especificação de requisitos</resumo>
  <data>12/05/2003</data>
  <acesso>0</acesso>
  <pessoas>
    <pessoa id="dgsante" papel="secretario" />
    <pessoa id="pedro" papel="participante" />
  </pessoas>
  <pauta>
    <item descricao="problemas" id="0" />
  </pauta>
  <bg_slides textfile=
"http://localhost:8080/autore/medias/meet/1/19/background/Slide_Texto.xml">
    <bg_slide href=
"http://localhost:8080/autore/medias/meet/1/19/background/Slide1.JPG" id="0" />
    <bg_slide href=
"http://localhost:8080/autore/medias/meet/1/19/background/Slide2.JPG" id="1" />
    <bg_slide href=
"http://localhost:8080/autore/medias/meet/1/19/background/Slide3.JPG" id="2" />
    <bg_slide href=
"http://localhost:8080/autore/medias/meet/1/19/background/Slide4.JPG" id="3" />
    <bg_slide href=
"http://localhost:8080/autore/medias/meet/1/19/background/Slide5.JPG" id="4" />
    <bg_slide href=
"http://localhost:8080/autore/medias/meet/1/19/background/Slide6.JPG" id="5" />
    <bg_slide href=
"http://localhost:8080/autore/medias/meet/1/19/background/Slide7.JPG" id="6" />
  </bg_slides>
  <sessoes />
  <ata />
  <midias>
    <midia href="http://localhost:8080/autore/medias/meet/1/19/video.mpg"
description="Video do cliente" />
    <midia href="http://localhost:8080/autore/medias/meet/1/19/modelo.doc"
description="Modelo de Documento de Requisitos" />
  </midias>
  <urls />
</reuniao>

```

Coleção: db/meet/sala

```
<?xml version="1.0"?>
<sala id="1" nome="Projeto Intermedia">
  <reunioes>
    <reuniao id="0" />
    <reuniao id="1" />
    <reuniao id="13" />
    <reuniao id="17" />
    <reuniao id="20" />
    <reuniao id="21" />
    <reuniao id="22" />
    <reuniao id="23" />
  </reunioes>
</sala>
```

Coleção: db/session

```
<?xml version="1.0"?>
<session id="136" type="2" date="2002.09.26 AD at 10:11:44 BRT">
  ...
  <refml>
    <ref id="11" date="Sat Mar 08 11:52:29 GMT-03:00 2003">
      <reforig type="session" time="0" pointer="/session[@id=5]" />
      <refdest type="session" time="0" pointer="/session[@id=4]" />
    </ref>
    <ref id="12" date="Sat Mar 08 11:57:00 GMT-03:00 2003">
      <reforig type="slide" time="0"
pointer="/session[@id=5]/whiteboardml/slide[@slide_id=7]" />
      <refdest type="audio" time="18.798"
pointer="/session[@id=4]/audioml/audio"/>
    </ref>
    <ref id="14" date="Sat Mar 08 11:58:45 GMT-03:00 2003">
      <reforig type="slide" time="0"
pointer="/session[@id=5]/whiteboardml/slide[@slide_id=1]" />
      <refdest type="slide" time="0"
pointer="/session[@id=4]/whiteboardml/slide[@slide_id=2]" />
    </ref>
    <ref id="15" date="Sat Mar 08 12:35:03 GMT-03:00 2003">
      <reforig type="session" time="0" pointer="/session[@id=5]" />
      <refdest type="session" time="0" pointer="/session[@id=4]" />
    </ref>
    <ref id="16" date="Sat Mar 08 12:38:49 GMT-03:00 2003">
      <reforig type="session" time="0" pointer="/session[@id=5]" />
      <refdest type="audio" time="7.774" pointer="/session[@id=4]/audioml/audio"/>
    </ref>
  </refml>
  <whiteboardml>
    ...
    <slide slide_id="2">
      <title>Slide 2</title>
      <url>http://localhost:8080/autore/medias/2o_2002/SCE-185/2/slide2.jpg</url>
      <text>
        <sentence>Novos rumos</sentence>
        <annotation>
          novas descobertas
          pesquisas recentes
        </annotation>
      </text>
    </slide>
    ...
  </whiteboardml>
</session>
```


Apêndice B – Códigos das *interfaces* Java criadas

XMLNativeDB.java

```
/* Generated by Together */

package packDB;

import java.lang.Exception;
import org.apache.xml.serialize.XMLSerializer;
import org.xmldb.api.modules.XPathQueryService;
import java.lang.System;
import org.xmldb.api.base.ResourceSet;
import org.apache.xml.serialize.OutputFormat;
import java.util.Arrays;
import packDB.ResultSet;
import org.xmldb.api.modules.XMLResource;
import org.xmldb.api.base.ResourceIterator;
import packDB.Xindice;
import java.util.Vector;
import java.lang.String;
import java.io.StringWriter;
import org.w3c.dom.Node;
import org.w3c.dom.Document;
import java.lang.Integer;
import org.xmldb.api.modules.XUpdateQueryService;
import org.xmldb.api.base.XMLDBException;
import org.w3c.dom.Element;

public interface XMLNativeDB {
    /**
     * Run Update against one XML document stored in database
     * @param xupdate: the XUpdate query string to run against a XML collection
     * @param id: the resource ID of the document
     */
    long updateResource(String id, String xupdate) throws Exception;

    /**
     * Run Queries against collections of XML documents stored in database
     * @param query: the XPath query string to run against a XML collection
     */
    ResultSet query(String xpath) throws Exception;

    /**
     * Retrive an entire document from database, as a DOM tree
     * @param docname: name of the document to retrieve
     */
    Document getDocumentAsDOM(String docname) throws Exception;

    /**
     * Formats a document in database.
     * @param id: the document id
     */
    void formatResource(String id);

    /**
     * Saves a DOM tree in database as a XML document
     * @param doc: the DOM node that contains the document
     * @param id: the unique ID for the document
     */
}
```

```

    */
    void saveDOM(Node doc, String id) throws Exception;

    /**
     * Retrives an entire Document from database, as text.
     * @param docname: name of the document to retrieve
     */
    String getDocument(String docname) throws Exception;

    /**
     * Run Updates against collections of XML documents stored in database
     * @param xupdate: the XUpdate query string to run against a XML collection
     */
    long update(String xupdate) throws Exception;

    /**
     * Get the next value of the property "id" of a given node
     * @param element: the name of the node to retrieve next id
     */
    String getNextId(String element) throws Exception;
}

```

Captureable.java

```

/* Generated by Together */

package packDB;

import packUtil.ZipUtil;
import java.util.Arrays;
import packDB.AulaBean;
import org.w3c.dom.Element;
import org.w3c.dom.Document;
import java.util.Vector;
import java.lang.Long;
import packDB.TurmaBean;
import javax.xml.parsers.DocumentBuilder;
import packDB.ResultSet;
import java.lang.String;
import java.lang.Integer;
import javax.xml.parsers.DocumentBuilderFactory;
import java.lang.Exception;

/**
 * This inteface defines the common methods of captureable session contexts
 * (Classes, Meetings, Conferences, etc)
 * @author Daniel Sante
 * @version 1.0
 */
public interface Captureable {
    /**
     * This method updates the new slides data based on the given order.
     * @param slides: the new slides data.
     * @param order: the corresponding order of appearance for each slide.
     */
    String updateOrderedSlides(Vector slides, String[] order) throws Exception;

    /**
     * Return all inserted extra medias.
     * @param id: id of the resource
     */
    ResultSet selectMedias(String id) throws Exception;
}

```

```

/**
 * This method returns the resource date
 * @param id: the resource id.
 */
String getDataById(String id) throws Exception;

/**
 * This method inserts a new URL in given resource
 * @param id: the resource id.
 * @param href: address
 * @param desc: description of the URL
 */
String addURL(String id, String href, String desc) throws Exception;

/**
 * This method returns all extra visited URLs of the resource
 * @param meetid: the resource id.
 */
ResultSet selectURLs(String meetid) throws Exception;

/**
 * This method returns all captured session for a given resource
 * @param id: the resource id.
 */
ResultSet selectSessions(String id) throws Exception;

/**
 * This method removes a media from given resource
 * @param id: the resource id.
 * @param href: location of the media file
 */
String delMedia(String id, String href) throws Exception;

/**
 * This method updates the background slide data in the database.
 * @param slides: vector containing all background slide data.
 * @param http: href to retrieve the slides over WWW
 */
String updateBgSlidesXML(Vector slides, String http) throws Exception;

/**
 * This method returns the number (order) of a given session of the resource
 * @param id: the resource id.
 * @param sessid: the id of the session.
 */
String numberSession(String id, String sessid) throws Exception;

/**
 * This method saves the background slide files.
 * @param zipfile: zip archive containing all slide image files, plus the
slide XML text descriptor
 * @param path: the path to decompress & save the image files
 */
Vector saveBgSlides(String zipfile, String path);

/**
 * This method creates a new resource instance in the database.
 */
String create();

/**
 * This method returns one specific background slide of the resource

```

```

        * @param meetid: the resource id.
        * @param slideid: the id of the slide.
        */
        ResultSet selectBgSlide(String aulaid, String slideid) throws Exception;

    /**
     * This method returns all resource ids that haven't captured sessions in the
     database
     */
        Vector getNotCap() throws Exception;

    /**
     * This method returns the resource code of given captured session
     * @param sessid: the session id.
     */
        String getIdBySession(String sessid) throws Exception;

    /**
     * This method inserts a new media in given resource
     * @param id: the resource id.
     * @param href: location of the media file
     * @param desc: description of the media file
     */
        String addMedia(String id, String href, String desc) throws Exception;

    /**
     * This method returns the resource title
     * @param id: the resource id.
     */
        String getTituloById(String id) throws Exception;

    /**
     * This method returns the resource id
     */
        String getId();

    /**
     * This method removes an URL from given resource
     * @param id: the resource id.
     * @param href: address
     */
        String delURL(String id, String href) throws Exception;

    /**
     * This method returns the resource background slides
     * @param id: the resource id.
     */
        ResultSet selectBgSlides(String id) throws Exception;

    /**
     * This method sets the resource id
     * @param id: the resource id.
     */
        void setId(String id);
}

```

Apêndice C – Estrutura de arquivos do AutorE

Diretório: autore/ (raiz do http): contém os *scripts* JSP e páginas dos módulos do sistema.

Arquivos do AutorE-Class (formulários e scripts):

Módulo de Preparação

- *newclass.jsp*: formulário de criação de nova aula;
- *newclass-act.jsp*: script de criação de uma nova aula;
- *newclass2.jsp*: formulário de seleção de slides para reuso;
- *newclass2-act.jsp*: script de inserção e formulário de ordenação dos slides;
- *newclass2-end.jsp*: script de finalização.

Módulo de Referência

- *selclassref.jsp*: formulário de escolha de aulas origem e destino para referência;
- *listclassref.jsp*: lista as referências definidas para uma sessão;
- *refclass.jsp*: formulário de definição de referências entre sessões.

Módulo de Expansão

- *selclass.jsp*: formulário de escolha de disciplina / turma;
- *expclassslide.jsp*: formulário para escolha de slide;
- *expclassslide2.jsp*: formulário para inserção de texto/anotações no slide;
- *expclassslide-act.jsp*: script de inserção de texto/anotações no slide;
- *expclassata.jsp*: formulário para inserção de tarefas na aula;
- *expclassata-act.jsp*: script de inserção de tarefa;
- *expclassmedia.jsp*: formulário para inserção de uma mídia na aula;
- *expclassmedia-act.jsp*: script de inserção de mídia;
- *expclassurl.jsp*: formulário de inserção de uma URL na aula;
- *expclassurl-act.jsp*: script de inserção de URL.

Arquivos do AutorE-Meet:

Módulo de Preparação

- *newmeet.jsp*: formulário de criação de uma nova reunião;
- *newmeet-act.jsp*: script de criação de uma nova reunião;
- *newmeet2.jsp*: formulário de definição de pauta e participantes;
- *newmeet2-act.jsp*: script de inserção de pauta e participantes;
- *newmeet3.jsp*: formulário de seleção de slides para reuso;
- *newmeet3-act.jsp*: formulário de ordenação de slides;
- *newmeet3-end.jsp*: script de inserção de slides;
- *newmeet-end.jsp*: script de finalização.

Módulo de Referência

- *selmeetref.jsp*: formulário de escolha de reuniões origem/destino para referência;
- *listmeetref.jsp*: lista as referências definidas para uma sessão;
- *refmeet.jsp*: formulário de definição de referências entre sessões.

Módulo de Expansão

- *selmeet.jsp*: formulário de escolha de sala;
- *expmeetslide.jsp*: formulário para escolha de slide;
- *expmeetslide2.jsp*: formulário para inserção de texto/anotações no slide;
- *expmeetslide-act.jsp*: script de inserção de texto/anotações no slide;
- *expmeetata.jsp*: formulário para inserção da ata da reunião;
- *expmeetata-act.jsp*: script de inserção da ata;
- *expmeetmedia.jsp*: formulário para inserção de uma mídia na reunião;
- *expmeetmedia-act.jsp*: script de inserção da mídia;
- *expmeeturl.jsp*: formulário para inserção de uma URL na reunião;
- *expmeeturl-act.jsp*: script de inserção de URL.

Arquivos comuns aos contextos:

- *index.jsp*: página inicial do sistema;
- *menuprep.jsp*: menu sensível a contexto do Módulo de Preparação;
- *menuref.jsp*: menu sensível a contexto do Módulo de Referência;
- *menuexp.jsp*: menu sensível a contexto do Módulo de Expansão;
- *pop_slide.jsp*: janela pop-up que exibe um slide de preparação;
- *pop_slide_cap.jsp*: janela pop-up que exibe um slide capturado;
- *delitem.jsp*: script genérico de remoção de itens em documentos XML;
- *selcontext.jsp*: página de seleção do contexto do sistema;
- *setcontext.jsp*: script que alterna os contextos do sistema;
- *welcome.jsp*: página de boas-vindas;
- *pop_audio.jsp*: janela pop-up que contém o applet para reprodução de áudio/vídeo;
- *PlayerApplet.class*: applet que reproduz áudio e vídeo;
- *PlayerApplet.java*: código-fonte do applet;
- *login-act.jsp*: script que realiza o login de um usuário do sistema;
- *ref-act.jsp*: script que insere referências entre sessões;
- *logout.jsp*: script que realiza o logout de um usuário do sistema.

Diretório: autore/config: contém o arquivo de configuração do sistema.

- *config.jsp*: define os diretórios e endereço-base da instalação do sistema.

Diretório: [autore/layout](#): contém as definições de layout do sistema, como cabeçalhos e rodapés das páginas.

Arquivos comuns aos contextos:

- *header.jsp*: cabeçalho padrão do sistema;
- *topmenu.jsp*: menu superior, sensível a contexto;
- *footer.jsp*: rodapé padrão do sistema;
- *layout.css*: arquivo de folha de estilo com as definições de fontes, cores, etc.;
- *header-prep.jsp*: cabeçalho padrão do Módulo de Preparação;
- *header-ref.jsp*: cabeçalho padrão do Módulo de Referência;
- *header-exp.jsp*: cabeçalho padrão do Módulo de Expansão;
- *permission.jsp*: verifica permissão/login de usuários (incluído no header).
- *powered.jsp*: cabeçalho com os logotipos *powered by*.

Além desses, existem ainda mais três diretórios: o diretório [autore/javadoc](#) contém a documentação da Camada de Acesso do sistema em formato HTML, gerado pelo aplicativo *javadoc*; o diretório [autore/medias](#) armazena os arquivos de mídia enviados através de *upload* e o diretório [autore/images](#) contém as figuras usadas pelas páginas do sistema.