UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação
ISSN 0103-2569

Automatically Linking Live Experiences Captured through a Ubiquitous Infrastructure

Alessandra Alaniz Macedo
Laércio Augusto Baldochi Jr.
José Antonio Camacho-Guerrero
Maria da Graça C. Pimentel

Nº 211

RELATÓRIOS TÉCNICOS

São Carlos – SP
Set./2003

Automatically Linking Live Experiences Captured through a Ubiquitous Infrastructure

Alessandra Alaniz Macedo, Laércio Augusto Baldochi Jr., José Antonio Camacho-Guerrero, Maria da Graça C. Pimentel

[1]Departamento de Ciências de Computação e Estatística
Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo
CP 668, 13560-970 - São Carlos/SP - Brazil

{ale,baldochi,jcamacho,mgp}@icmc.usp.br

Abstract. *The objective of investigation presented in this paper is to support the automatic linking among live experiences captured through C&A applications, towards providing information related to the captured experience accessed later by users. Then, we defined an infrastructure composed by a "document builder" to process captured information as XML documents, a "linking builder" to identify semantic relationships between the captured content, a "storage manager" to store the XML corresponding to the captured session and the hyperlinks identified, and a "presentation manager" to manipulate style sheets that allow users to access the captured content in a variety of presentation formats. To illustrate the results of our work, we present hyperlinks created between documents produced by iClass, a live lecture capture system build upon the capture module.*

1. Introduction

Authoring meaningful and useful multimedia documents — those that present effective integration between several media components — is an expensive task. Everyday experiences are rich in information that, if captured and made available as multimedia documents, may provide a valuable source of information. Moreover, humans need support in capturing the salient memories from live experiences that they can then call upon at a later time. There are many situations in our everyday lives that satisfy both of these criteria.

In the beginning of the 90's, Mark Weiser envisioned that computer applications could be embedded in the environment, aiding people without changing the way they perform their activities. This transparent integration of technology in the everyday life was called ubiquitous computing [Weiser, 1991]. Many applications exploit the ubiquitous computing paradigm to support the automatic capture of information in live experiences towards generating corresponding documents to be accessed during the capture session or later to visualize the results [Abowd, 1999, Davis et al., 1999, Mukhopadhya and Smith, 1999, Mynatt et al., 1999, Richter et al., 2001].

In previous work, we have investigated the use of document-processing techniques and storage abstractions in order to support the development of capture and access (C&A) applications [Baldochi et al., 2003a]. To carry out this investigation, we built a

component-based infrastructure that offers C&A capabilities for a large variety of applications [Baldochi et al., 2003b]. To provide extra information to users at the end of capture sessions, C&A applications could be augmented by specialized automated services. Those services, outside of the live experience, can add information by highlighting relationships between the captured experiences.

For some years, researchers have presented studies that support the definition of relationships between textual information using Information Retrieval techniques [Salton and Allan, 1993, Blustein, 2000]. Some of those studies explore the manipulation of matching mechanisms such as lexical chains [Green, 1999] and Latent Semantic Indexing (LSI) [Furnas et al., 1988] to identify similarities between text elements in order to define hypertext links. In terms of defining similarities between documents, we have created linking services allowing the automatic identification of relationships in textual homogenous Web-based repositories based on different approaches: (a) using simple matching lexical algorithms [Pimentel et al., 2001]; (b) experimenting with semantic approach based on LSI [Macedo et al., 2001]; (c) considering an open hypermedia linkbase accessed via a Web interface [Macedo et al., 2002b]; (d) allowing user-interaction over the hyperlinks generated [Macedo et al., 2002a]; and (e) as a linking component of a recommender service [Macedo et al., 2003]. As another result of these investigations, we built a configurable service, named LinkDigger, in which up to five repositories can be analyzed and the relationships identified are stored in an open hypermedia linkbase accessed via a Web interface [Camacho-Guerrero et al., 2002a, Camacho-Guerrero et al., 2002b].

Towards integrating the capture and linking services, we are investigating the automatic creation of hyperlinks between XML documents that represent captured live experiences through a ubiquitous infrastructure. We define an infrastructure to be exploited by capture and access applications intending to present related captured information as hyperlinks in the Web. This infrastructure represents: (a) the automation of the process involved in the capture phase of typical C&A applications; (b) the processing and storage of captured information as XML documents; (c) the identification of semantic relationships between the captured content using a linking service that manipulates semantic as similarity of terms according to Latent Semantic Indexing theory [Macedo et al., 2002b] [Macedo et al., 2001]; (d) the storage of the associations identified in an open linkbase [Bulcão Neto et al., 2002]; and (e) the definition of style sheets that allow users to access the captured content for review in a variety of presentation formats by processing XML documents in a Cocoon pipeline [Baldochi et al., 2003b].

As a *result*, our infrastructure provides software support for automating the process of identification of relationships between documents corresponding to a live session. To illustrate those results, we present hyperlinks created between documents produced by iClass, a live lecture capture system build upon our capture infrastructure [Baldochi et al., 2003a].

The remainder of this paper is organized as follows. In Section 2., we briefly discuss the approaches of C&A infrastructures and linking services that we exploited in the work presented in this paper. In Section 3., we present the architectural details showing the integration of services in order to provide capture, access, linking, storage and visualization software support to C&A applications. In Section 4., we present a C&A application, the iClass System, used to illustrate our approach. In Section 5., we describe experiments to create hyperlinks between captured information from live lectures captured by iClass. We conclude with some discussion

and future works in Section 6..

2. Background: Starting points

In order to investigate the creation of hyperlinks using documents that represent live experiences captured by means of ubiquitous applications, we have exploited and extended our previous work.

In this section, we present a set of services that offer infrastructural support for the capture and access of human activities such as classes and meetings. After that, we introduce linking services which are able to create links between the semantic related captured content.

2.1. INCA: Network Abstractions

Capture and access applications are usually composed by a confederation of devices and software systems, presenting a inherent distributed nature. Truong & Abowd developed an INfrastructure for Capture and Access Applications (INCA) in order to provide network abstractions demanded in the implementation of those applications [Truong and Abowd, 2002].

INCA provides architectural support for building applications by means of a set of key design concerns: (a) *Capture* (part of the system is responsible for the capture of information as streams of data); (b) *Storage* (part of the system is responsible for the storage of attribute-tagged information); (c) *Transduction* (when information needs to be converted into different formats and types, part of the system must transduce the information); and (d) *Access* (part of the system provides synchronous access to multiple integrated streams of information that are gathered as a response to context-based queries).

For each design concern, INCA implements a module and an interface that developers must instantiate and implement when building applications. The basic entities are the C&A modules, which behave like producers and consumers in a peer-to-peer approach. In order to automatically control the presence of C&A modules, INCA provides a *Registry* that manages publishing and subscribing operations done by application modules. When published attributes and subscribed attributes match, INCA enables the communication flow between the corresponding capture and access modules. INCA treats data as large binary objects. Information is assembled as *DataObject*s tagged with a list of name-value attributes that associates meta information with the data. By imposing this uniform representation to data, INCA avoids that portions of applications have to be re-implemented when new data types have to be supported.

2.2. xINCA: Functionality Abstractions

Besides architectural aspects, a large number of C&A applications present recurrent functionalities. A typical example occurs in the educational domain: most of the applications developed for this domain needs software to control interactions in a whiteboard. Other recurrent functionalities are also present in C&A applications: chat, audio, video and weblog are some examples. These functionalities can appear in any combination and suggest a component-based approach. With this in mind, we developed an extended INfrastructure for Capture and Access Applications (xINCA) as a layer on top of INCA, making use of its modules and abstractions.

xINCA provides software components that captures user-interactions with electronic whiteboards and PDAs, text generated during chat sessions, URLs visited during Web browsing as well as streams of audio and video recorded during live capture sessions.

xINCA components implement instances of INCA modules (capture modules, access modules or both capture and access modules) and are registered at runtime in a INCA *Registry* entity. xINCA components communicate using a session identifier — a *session* represents a period of interaction between components and has a unique identifier as long as it lasts. Once sharing the same session identifier and functionality type, components from a given C&A application are able to communicate and exchange information with any other component from any other C&A application. Moreover, xINCA components can be added or removed at any moment during a session. Figure 1 depicts the communication flow between components in two simultaneous sessions.

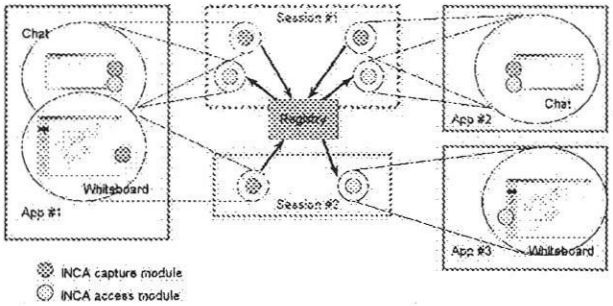

Figure 1: Communication flow between xINCA components in two simultaneous sessions [Baldochi et al., 2002]

In Figure 1, App #1 is a C&A application with a chat component and a whiteboard component, App #2 is a capture and access application with a chat component and App #3 is an access-only application with a whiteboard component. The chat components of App #1 and App #2 are registered under the same session identifier (Session #1) and, consequently, have an active communication channel that allows them to exchange information. Similarly, the whiteboard components of App #1 and App #3 are registered under a same session identifier (Session #2) so that they can also exchange data. However, the communication flow among App #1 and App #3 is unidirectional since App #1 captures information in the whiteboard while App #3 only access the information captured by App #1.

2.3. Linking Services

Also in previous work, we have investigated mechanisms allowing the automatic identification of relationships in textual homogenous Web-based repositories by using simple matching lexical algorithms, having experimented with the integration of two complementary repositories supporting a single graduate course [Pimentel et al., 2001]. When experimenting with LSI considering the same repositories, the answer set was bigger, which is explained by the homogeneity of the repositories and the fact that a manual normalization of the vocabulary was performed [Macedo et al., 2001].

As a result of our investigation with LSI, we built a service, named LinkDigger, in which up to five repositories can be analyzed, storing the identified relationships in an open hypermedia linkbase accessed via a Web interface [Macedo et al., 2002b]. LinkDigger was extended to allow user-interaction over the semantic links generated [Macedo et al., 2002a], and as a linking component of a recommender service [Macedo et al., 2003]. In the next section, we show our linking service also being used as a linking component. Although in this paper, LinkDigger will be relating captured information produced by a C&A application.

3. The Infrastructure

By integrating and deploying a set of different services in a single infrastructure we were able to address many requirements of C&A applications, making it easier to create those applications, as high level APIs are provided for the application developers.

In this section, we present the architecture of our infrastructure, detailing the integration of services that provide capabilities for capturing, storing, retrieving, linking and accessing information that represents live experiences.

According to Figure 2, our infrastructure comprises four main modules: (a) Document Builder, (b) Storage Manager, (c) Linking Builder and (d) Presentation Manager. Those modules are presented in the following subsections.

3.1. Document Builder

The starting point for the development of our infrastructure was the modeling of the information produced by C&A applications. Based on the analysis of use cases, we modeled each session as an XML document. Although this approach was mainly motivated by the need to exchange information between the many modules that comprise our infrastructure, it also facilitates the interchange of information among third-part applications.

In order to define the structure and the content of the documents that represent captured sessions, the first step was to define data schemas for all xINCA components (whiteboard, video, audio, chat and weblog). After that, we defined a *session schema*, which aggregates the schemas of xINCA components together with a *contextual schema*, where we modeled the contextual information related to a session.

Figure 3 presents the *session schema*, where all xINCA components are integrated. As

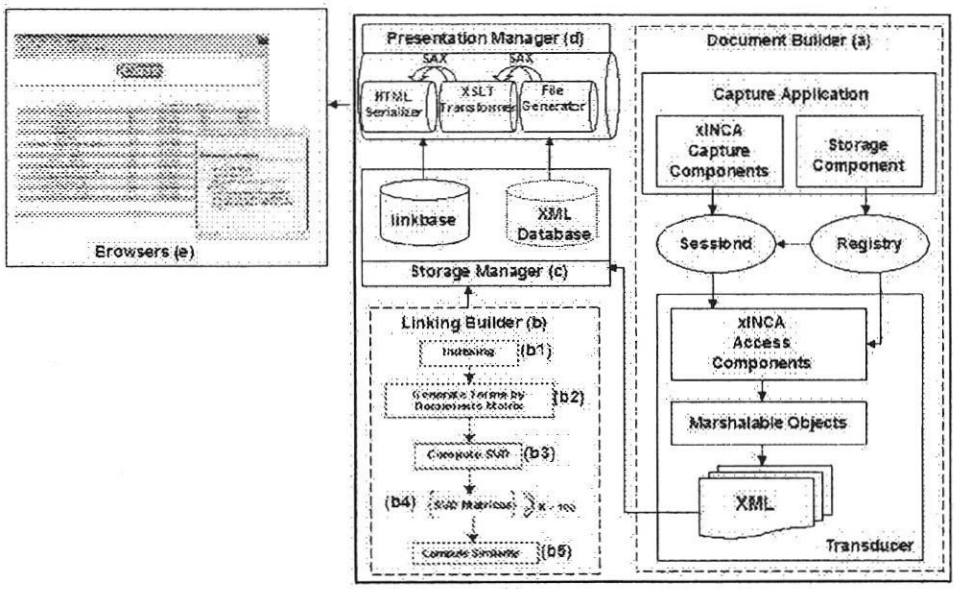

Figure 2: Infrastructure to support capture and access applications intending to present related captured information as hyperlinks.

this schema is too large, we show the contents of the *contextual schema* in the second column. The complex type *ContextInfo*, shown in the right column, models the application's contextual information. Truong et alli have defined five dimensions that contextualize a C&A application: (a) *who* are the users during capture and access; (b) *what* is captured and accessed; (c) *when* and (d) *where* does capture and access occur, and (e) *how* is capture and access performed. The first four cited dimensions appear as sub-elements of *context_info* [Truong et al., 2001]. The fifth dimension (*how*) is represented by the *type* attribute of the element *session*.

The *session schema* was then used as input for a data binding mechanism called JAXB – Java Architecture for XML Binding [Lin, 2002]. JAXB compiles each schema into a set of Java classes. The generated classes handle all the details of XML parsing and formatting, allowing the implementation of applications that can read, manipulate and create XML documents without the need to write any logic to process the documents' elements and attributes.

The data binding mechanism compiles our schemas generating all classes needed to process the data produced during the capture of a live experience. Besides the classes produced by JAXB, it was necessary to write the class that handles the transduction of each component's data into XML documents — called *Transducer*.

The *Transducer* instantiates correspondent access components for the capture components present in the application. It also instantiates objects from the classes generated by JAXB. Using the access components, the *Transducer* receives all the information produced by the application's capture components and transduces this information to the objects provided by the data binding mechanism.

The *Document Builder* works in a client-server fashion. For the application developer perspective, the implementation of the data management part of a C&A application follows a

Figure 3: The *session schema* (left column), which aggregates the schemas of xINCA components with a *contextual schema* (right column)

straightforward procedure: all she needs to do is to instantiate a *Storage Component* object, passing a set of parameters that includes a session identifier and a constant that tells the *Transducer* which access components it must instantiate in order to receive the data produced by the application.

Two daemons play an important role in the architecture of the *Document Builder* Module: *Sessiond* and *Registry*. The first one was built in order to support the capture of distributed sessions, allowing the construction of applications where remote peers are added and removed during runtime (further details about *Sessiond* can be found in [Baldochi et al., 2003a]). The *Registry* is the INCA daemon responsible for providing transparent communication between client and server side of applications. It also provides a callback mechanism to notify the status of its connections. Using this mechanism, *Sessiond* is able to determine the end of a session and call the *stop()* method in the appropriate *Transducer*, causing the generation of the document that represents the session.

3.2. Linking Builder

In order to create hyperlinks from captured information, the *Linking Builder* manipulates the XML documents stored by the *Transducer* on an XML database, part of the *Storage Manager* presented next.

The *Linking Builder* automatically generates hyperlinks between live experience information captured and recorded by the *Capture Application* supported by the *Transducer*. After receiving the XML documents corresponding to each session captured, the *Linking Builder* performs two main tasks: indexing of all XML documents and creation of links among them according to the Latent Semantic Indexing method (LSI) [Furnas et al., 1988]. LSI is used to overcome the problems related to the use of lexical based analysis. The similarities defined by LSI are based on closeness of terms in a semantic space built according to co-occurrence of all terms in collection of documents manipulated instead of matching.

The underlying processing of the *Linking Builder* is as follows:

- *Indexing*. Initially all XML documents are indexed, i.e., significant words (excluding stop words) are extracted from the XML documents (see Figure 2(b1)). We used the mnoGoSearch [Group, 2001] general public license search engine to extract words from the XML dcouments. Since many words are extracted, it is important that, at the time of the indexing, the words be given an appropriate weight in terms of the number of times they appear in a given document relative to the number of times that they appear in each document in the repositories of documents. In order to attribute appropriate to the XML documents manipulated, we have used a term-weighting scheme presented in [Salton and Buckley, 1988].
- *Generate Terms by Documents Matrix*. The index resulting from the previous step is used to generate a term by document matrix (see Figure 2(b2)). This term by document matrix is called matrix X and it is exploited by Latent Semantic Indexing [Furnas et al., 1988].
- *Compute SVD*. The matrix X is decomposed into the product of three other component matrices T, S and D' using Single Value Decomposition (SVD) which is part of LSI theory (see Figure 2(b3)). Following the decomposition by SVD, the k most important dimensions (those with the highest values in the singular matrix S) are selected. All

other factors are omitted. The amount of dimensionality reduction, i.e., the choice of k, is critical and is an open issue in the literature. Ideally, k should be large enough to fit the real structure in the data, but small enough such that noise, sampling errors and unimportant details are not modeled. Generally we have considered k as 200 as suggested in the literature by some researchers [Deerwester et al., 1990].

- *SVD Matrices*. A semantic matrix is generated by the computation of the inner-product among each column of the matrix generated on the last step (see Figure 2(b4)). Those columns represent XML documents generated by the *Document Builder* when a capture component is enabled.
- *Compute Similarities*. Given the semantic matrix generated in the previous step, relationships between XML documents are identified by considering the cells that have the higher values of similarity (see Figure 2(b5)). A threshold level of similarity is used to filter the links created to generate a relevance semantic matrix which is used to identify semantic links between documents. The links generated are stored at the *Storage Manager* (see Figure 2(c)).

We have developed the *Linking Builder* to be reusable in other implementations. Then, we define a configuration file to specify the context within which it will be employed. This configuration file describes information such as the vocabulary, stop list and source of URLs.

3.3. Storage Manager

After generating the XML documents and their respective hyperlinks, our infrastructure must provide a way to manage both of them, providing an easy way for applications to store and retrieve content.

During the design of our infrastructure, we opted for defining one storage repository to be accessed during the capture session, called *Storage Component*, and another, called *Storage Manager*, to be exploited by the *Presentation Manager* and some augmenting mechanisms such as the *Linking Builder*.

The *Storage Manager* provides an API that allows applications to store, retrieve, query and extend document. Using this API, we are implementing a presentation mechanism that retrieves XML documents and process them on-the-fly, generating documents in different presentation formats (see Section 3.4. for further details).

The *Storage Manager* is composed by an XML document repository and an open hypermedia linkbase. Both are presented in the next two subsections.

3.3.1. An XML database

In order to provide ubiquitous data access to C&A applications we built a distributed document repository using the facilities provided by eXist [Meier, 2003], a native XML database. eXist functionality is based on hierarchical collections, similar to the storage process in a file system. Besides the facilities for creating and removing collections, storing and retrieving documents, eXist also provides a query mechanism based on XPath.

As it is highly desirable to have a repository available in the same LAN as its client applications and considering the inherent distributed nature of C&A applications, we decided to build

a distributed storage service. Our strategy to build a distributed repository consists in brokering the communication between applications and their local repository. This was done by replacing the XML-RPC interface of eXist by an entity called *Document Manager*, which is an extension of the XML-RPC server provided by eXist. According to this extension, all method calls that change the state of the local repository are spread to all other active repositories. Details about the implementation of the distributed repository is available in [Baldochi et al., 2003b].

3.3.2. An open linkbase

In the proposed infrastructure, the hyperlinks created by the *Linking Builder* are been stored on an open hypermedia linkbase. Nowadays, we have been used a linkbase, called WLS [Bulcão Neto et al., 2002], which offers some facilities that common databases do not offer.

The WLS linkbase was developed as an API, so that application developers can reuse and combine the available operations with their own building blocks. By means of its API, WLS can be reused in different contexts, reducing the authoring effort as shown in [Bulcão Neto et al., 2002].

During the design of the infrastructure, we defined a mapping between objects manipulated by the infrastructure and the classes of the WLS presented in Figure 4. For example, Anchor is a set of terms of each XML document generated after capture session and EndPoints are equivalent to terms mapped to Anchors. Link is a set of two EndPoints identified to relationships defined by *Linking Builder*. Context is a collection of hyperlinks generated automatically. Nodes corresponds to each XML document that has been found to be similar to another document. Finally, Semantics correspond to pairs of similar terms. These pairs of terms were automatically defined by the linking service and stored on the WLS linkbase, independently of the documents identified holding the information.

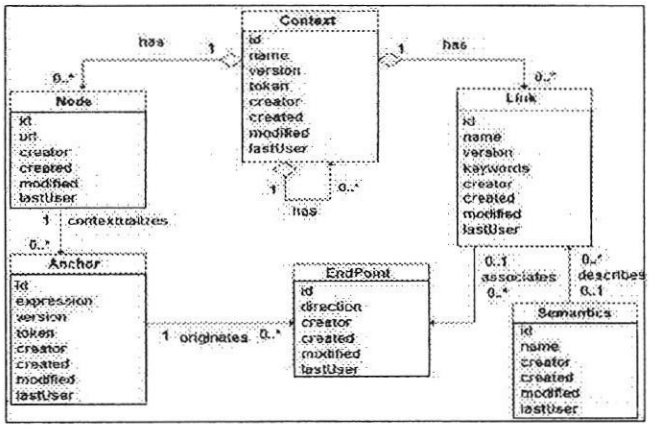

Figure 4: Conceptual Model of WLS linkbase [Bulcão Neto et al., 2002].

Because of the relationships among the Anchor, EndPoint and Link classes, WLS provides support to n-ary multidirectional links and to the sharing of an anchor between several links endpoints to our infrastructure.

Besides WLS tables and XML documents, the *Storage Manager* manipulates additional tables that are used by the *Linking Builder* such as the stop words list and the weights of terms.

The *Presentation Manager* obtains information from the containers of the *Storage Manager* to present it as Web pages to users.

3.4. Presentation Manager

In C&A applications, the access phase allows the visualization of information being captured during a live experience. Otherwise when the capture session is finalized, visualization implies that users may both see and interact with the presentation document supplied by the *Presentation Manager*.

Considering the diversity of existing C&A applications and platforms, the presentation of documents must be possible in a variety of formats — so as to conform to multimedia characteristics and user preferences, for instance. Overall, our infrastructure aims at supporting the documents to be accessed via the Web.

Given the importance of the degree of automation provided by a C&A application so that the information captured results in appropriate documents, it is paramount that, in the context of our work, the presentation documents be produced as a result of the processing of XML documents generated at the end of a live session. This scenario makes the use of XSLT transformations very suitable for the automatic generation of presentation documents. In fact, the generation of presentation documents defines a pipeline: an XML document corresponding to a captured session with links is processed exploiting an XSLT style sheet that generates a presentation document in a format such as SMIL. The underlying processing of the *Presentation Manager* is presented in Figure 2(d).

Our infrastructure provides a set of style sheets that process the XML documents corresponding to a captured session with its links and generates presentation documents in SMIL, XHTML+SMIL and HTML. Different versions of each style sheet allow conformity to different media stream servers and other formats can be added as needed.

In the case that the presentation documents are processed on-the-fly during the access phase, the documents are generated on demand and can be updated. However, the performance in the generation is an important point to be considered since the response to the user request must occur at a reasonable time. In our infrastructure, we exploit the Cocoon Java-based Framework [Apache, 2002]. Cocoon uses the concept of pipelines to process XML documents: the documents advance in the pipeline by means of SAX events. It has allowed, with an acceptable performance, the generation on-the-fly of a variety of presentation formats as a result of the processing of XML documents by means of XSLT transformations.

Each stage of the Cocoon pipeline is modelled with a particular type of component. For instance, the input can be modelled with the *File Generator* component associated with a session XML document. Next, the processing can be modelled with an *XSLT Transformer* component associated with a style sheet that generates HTML. At the end of the pipeline, the output can be modelled with a *HTML Serializer* component that serializes the presentation document in HTML, the final result of the overall processing. This example describes the Cocoon pipeline used to process the XML documents from the live sessions with the many style sheets provided by our infrastructure. As response to a user request, the correct pipeline is identified for presenting the appropriate document.

Figure 2(e) presents small sample documents generated on-the-fly, via a Cocoon

pipeline, in the context of iClass, a C&A application, presented in the next section.

4. The iClass System

We have developed a C&A application, the iClass System [Baldochi et al., 2003a], which we will use throughout the rest of this paper to illustrate our approach. To the proposed work, iClass was augmented by the *Linking Builder* module to support the creation of hyperlinks between captured information.

The iClass system is an improved version of the eClass[1], which is able to record several pieces of information produced during the lecture, including strokes and slides from an electronic whiteboard, audio from a cordless microphone attached to the lecturer, video from a inexpensive Web camera located in a fixed position in the classroom and Web pages visited from the lecturer's Web browser. As a result, at the end of the lecture, a XML document integrating the different captured media is produced by the *Transducer* and automatically stored on a document repository, which is part of the *Storage Manager*.

The iClass System makes extensive use of xINCA's abstractions and *Document Builder* module. In our integrated service, the *Linking Builder* accesses the document repository to get information to be related. Nowadays, we are using the eXist database to store the XML documents and the LinkDigger service to execute the linking process.

The core of the iClass system consists basically of a Java applet comprising one or more capture components and a storage component. The capture components, one for each functionality (whiteboard, audio, video and web logging), are instantiated in the beginning of the lecture according to the instructor's choice or available devices. The storage component is responsible for contacting a server informing which capture components are running on the client. Then, the storage service instantiates a *Transducer* object containing the corresponding access components, which will receive the information produced by the capture components running at the client applet. Using a callback mechanism, access components deliver the received information for being stored by the *Transducer*. Components of a specific session (lecture) are registered under the same session identifier allowing several sessions to happen simultaneously.

The results of the automatic generation of hyperlinks considering live experiences captured through the iClass system are presented next.

5. Evaluation

After augmenting the iClass system towards automatically linking live experience captured through this system, we defined two experiments to evaluate this extension.

Initially, the (*Document Builder*) transformed all lectures captured, during the last five years, through the previous eClass application into XML documents. After that, those documents were stored by the *Storage Manager* on an XML database. At the present time, we are using the eXist database.

In order to define the hyperlinks, the *Linking Builder* reads all XML documents through the *Storage Manager* and applies the processing linking, previously presented. The *Linking*

[1]The eClass system was formerly known as Classroom 2000, developed at Georgia Tech [Abowd, 1999]

Builder module was extended by a filter mechanism to ignore image, audio, video and strokes files. Although we faced problems with the quantity to be manipulated because the major part of captured information in C&A applications comes from audio and video files. Those files would not be useful for a textual linking manipulation.

On the iClass system, more textual information could be created with manual transcription of handwriting activities over whiteboard. Although, professors do not have spent a lot of time on such task. From all captured lectures, we got just 840 slides with handwriting transcriptions. Then in the experiments presented in this paper, we are basically working with words extracted from titles of the captured lectures and some handwriting transcriptions.

To evaluate our infrastructure, we have exploited the captured lectures to define hyperlinks between them. Due to space restrictions, just two experiments are presented.

5.1. The First Experiment

In the first evaluation, we experimented our approach with all captured lectures manipulated by the iClass system. The totalized 1833 lectures represented by 3666 XML document extracted from 116 courses. Some of those 3666 XML documents are audio and video components of capture sessions.

After filtering the XML documents, the *Linking Builder* manipulated just 1833 XML documents (one per lecture) and generated a matrix X with 3036 lines (representing significant words) and 1833 columns (representing XML documents). To compose the matrix X, those words were filtered considering a stop word dictionary in order to manipulate just significant words. After that, the matrix X was processed according to the LSI technique. Our service considered k as 200 during the reduction of the matrices generated by SVD. Just 200 highest values in the singular matrix S were selected.

The results of the linking process were 822 hyperlinks automatically created between all 1833 lectures (see Table 1). Despite existing non-relevant links, the results were considered good enough for academic purposes according to researchers interested in this approach.

Analyzing each hyperlink, we could observe similar lectures, given by different teachers on distinct terms, were preciously related. For instance, the "More Movies on Interaction" lecture given by Dr. Elizabeth Mynatt during Fall-99 was related to the Dr. Abowd's lecture called "Movies on Interaction" (See Figure 5(a)). Sometimes our approach also related lectures between different courses.

5.2. The Second Experiment

Since we had already experimented our approach over non-homogeneous repositories, i.e., different courses as Software Engineering, Robotics and Human Computer Interaction, we create a second experiment to verify the linking process just over HCI courses.

The HCI courses totalized 156 lectures, composed by 312 XML documents. After filtering 312 XML documents, the *Linking Builder* generated a matrix X with 1444 lines (significant words) and 156 columns (XML documents). Then, the matrix X was processed according to the LSI technique, which considered k as 100 during the reduction of the matrices generated by SVD.

Figure 5: The hyperlinks created between: (a) all lectures and (b) lectures given in HCI courses manipulated by the iClass system.

The results of the linking process were 41 hyperlinks automatically created between all 156 lectures (see Table 1).

For example, the "Human Computer Interface Design and Evaluation" course given by Dr. Gregory Abowd during Fall 2001 had various semantic relationships with a course about Human Computer Interface given by Dr. Elizabeth Mynatt and Dr. Wendy Newstetter during Spring 1999. Specifically, the "Contextual Inquiry" lecture from Abowd's course a good link was defined with the "Contextual Design" lecture of the later teachers. It can be considered a good hyperlink because Contextual Inquiry is part of Contextual Design. Probably this relationship would not be found in a lexical matching approach because the word contextual is an adjective, which would probably excluded during the indexing process.

Another example is the "Task Analysis" lecture given by Dr. Elizabeth Mynatt during Winter 1999, where our approach identified relations highly relevant with lectures given by Dr. Abowd during two different years, 1997 and 1998.

Figure 5 shows the hyperlinks created to the second experiment by the *Linking Builder* implemented as a Web application known as LinkDigger. Specifically Figure 5 (a) presents the hyperlinks created to the first experiment and Figure 5(b) shows hyperlinks between lectures manipulated in the second experiment.

Table 1: Characteristics of information manipulated in the evaluation of the service to create links among documents representing captured sessions

General Characteristics	Experiments	
	All Lectures	HCI Lectures
Number of documents manipulated	1833	312
Number of distinct terms	3036	1444
Number of stop words used	1619	1619
Number of hypertext links created	822	41

The two examples of our experimentation with the infrastructure proposed were chosen because the first presents a general scenario and the second one is about an area known by the

authors of this papers.

6. Conclusion and Future Work

The objective of investigation presented in this paper was to allow the automatic linking among live experiences captured through C&A applications, towards providing extra information to people, who access those captured experiences afterwards.

As a result, we defined an infrastructure, which provides software support to the automation of the process of identification of relationships between documents corresponding to a live session. Moreover, our infrastructure allows that solutions be reused, given the module-based infrastructure, and extended, given that a component can be modified if there is need for different functionalities, or new components can be added as needed — the requirement is that the new software follows the component-based document-oriented approach of the overall infrastructure.

Although, our investigations faced a serious problem — a lack of textual information in the C&A application context. During the evaluation of results, we could notice the captured lectures were basically composed by audio, video and stroke files. To get around this problem, we intend to develop an automatic transcription mechanism of pre-prepared slides and audio files. We are interesting in investigating the definition of hyperlinks among audio transcriptions. Also as future work, we intend to augment our infrastructure exploiting an open hypermedia Web system, which supports interactive authoring of multimedia material based on reusing of captured sessions [Sante, 2003].

7. ACKNOWLEDGMENTS

Alessandra Alaniz Macedo is a PhD candidate supported by FAPESP (99/115270). Maria Pimentel is supported by FAPESP and CNPq. Laércio Baldochi is a PhD candidate supported by CAPES and Fulbright. Maria Pimentel has international support to the InCA-SERVE Project by CNPq in Brazil jointly with Gregory Abowd who is supported by NSF in the U.S. We thank Gregory Abowd and Lonnie Harvel for their support.

References

Abowd, G. (1999). Classroom 2000: an experience with the instrumentation of a living educational environment. *IBM Systems Journal*, 38:508 – 530.

Apache (2002). The Apache Software Foundation: Apache Cocoon 2.0. URL:http://cocoon.apache.org/2.0/.

Baldochi, L., Cattelan, R., and Pimentel, M. (2003a). Building a middleware infrastructure for capture and access applications. In *Proceedings of XXX SEMISH - Seminário Integrado de Software e Hardware (to appear)*.

Baldochi, L., Cattelan, R., Pimentel, M., and Truong, K. (2002). Automatic generation of capture and access applications. In *Proceedings of the 8th Brazilian Symposium on Multimedia and Hypermedia Systems*, pages 100 – 115.

Baldochi, L., Pimentel, M., Andrade, A., Cattelan, R., and Sante, D. (2003b). Document engineering for capture and access aplications. In *Submitted for Proceedings of Document Engineering*.

Blustein, J. (2000). Automatically generated hypertext versions of scholarly articles and their evaluation. In *Proceedings of the Eleventh ACM Conference on Hypertext and Hypermedia*, pages 201–210.

Bulcão Neto, R. F., Izeki, C. A., Pimentel, M. G. C., Pontin, R. P. M., and Truong, K. N. (2002). An open linking service supporting the authoring of web documents. In *Proceedings of the ACM Document Engineering*, pages 66–73, Virginia-USA.

Camacho-Guerrero, J. A., Macedo, A. A., and Fortes, R. P. M. (2002a). Uma infra-estrutura configurável para serviços de criação automática de ligações. In *Anais do VII Brazilian Symposium on Multimedia and Hypermedia System*, pages 298 – 305, Fortaleza-CE,Brazil.

Camacho-Guerrero, J. A., Pimentel, M. G., and Bulcão Neto, R. F. (2002b). LinkDigger: um serviço aberto de criação automática de ligações semânticas. In *Anais do VII Brazilian Symposium on Multimedia and Hypermedia System*, pages 407 – 410, Fortaleza-CE,Brazi.

Davis, R., Landay, J., Chen, V., Huang, J., Lee, R., Li, F., Lin, J., Morrey, C., Schleimer, B., Price, M., and Schilit, B. (1999). NotePals: Lightweight note sharing by the group, for the group. In *Proceedings of the Conference on Human Factors in Computing Systems*, pages 338 – 345.

Deerwester, S., Dumais, S. T., Landauer, T., Furnas, G., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the Society for Information Science*, 41(6):391 – 407.

Furnas, G. W., Deerwester, S., Dumais, S. T., Landauer, T. K., Harshman, R. A., Streeter, L. A., and Lochbaum, K. E. (1988). Information retrieval using a singular value decomposition model of latent semantic structure. In *Proceedings of the 11th International Conference on Research & Development in Information Retrieval*, pages 465 – 480.

Green, S. (1999). Building hypertext links by computing semantic similarity. *IEEE Transactions on Knowledge and Data Engineering*, 11(5):713 – 730.

Group, M. (2001). Mnogosearch group. Internet. http://www.mnogosearch.ru.

Lin, T. W. (2002). Java architecture for XML binding (JAXB): A primer. Available in http://developer.java.sun.com/developer/technicalArticles/xml/jaxb/.

Macedo, A. A., Camacho-Guerrero, J. A., and Pimentel, M. G. C. (2002a). Incluindo abordagens de recuperação de informação em serviços de criação de hiperligações. In *XXVIII Conferencia Latinoamericana de Informática*, Montevideu,Uruguai.

Macedo, A. A., Pimentel, M. G. C., and Camacho-Guerrero, J. A. (2002b). An infrastructure for open latent semantic linking. In *Proceedings of ACM Hypertext 2002*, pages 107 – 116. ACM Press.

Macedo, A. A., Pimentel, M. G. C., and Guerrero, J. A. C. (2001). Latent semantic linking over homogeneous repositories. In *Proceedings of the ACM Symposium on Document Engineering*, pages 144 – 151. ACM Press.

Macedo, A. A., Truong, K. N., Camacho-Guerrero, J. A., and Pimentel, M. G. C. (2003). Automatically sharing web experiences through a hyperdocument recommender system. In *Proceedings of ACM Hypertext 2003*, page 9p. ACM Press. To appear.

Meier, W. (2003). eXist open source XML database. Available in: http://exist-db.org.

Mukhopadhya, S. and Smith, B. (1999). Passive capture and structuring of lectures. In *In Proceedings of the ACM Conference on Multimedia*, pages 477 – 487.

Mynatt, E. D., Igarashi, T., Edwards, W. K., and LaMarca, A. (1999). Flatland: New dimensions in office whiteboards. In *Proceedings of the Conference on Human Factors in Computing Systems*, pages 346 – 353.

Pimentel, M. G. C., Macedo, A. A., and Abowd, G. D. (2001). Linking homogeneous web-based repositories. In *Proceedings of International Workshop on Information Integration on the Web*, pages 35 – 42, Rio de Janeiro-RJ,Brazil. http://www.cos.ufrj.br/wiiw/schedule.html.

Richter, H., Abowd, G., Geyer, W., Fuchs, L., Daijavad, S., and Poltrock, S. (2001). Integrating meeting capture within a collaborative team environment. In *Proceedings of the Third International Conference on Ubiquitous Computing*, pages 123 – 138.

Salton, G. and Allan, J. (1993). Selective text utilization and text transversal. In *Proceedings of the ACM Conference on Hypertext 1993*, pages 131 – 144.

Salton, G. and Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513 – 523.

Sante, D. G. (2003). AutorE: suportando autoria evolucionária em ambientes de captura. Diss., Instituto de Ciências Matemáticas e de Computação da USP, São Carlos/SP.

Truong, K. and Abowd, G. (2002). Enabling the generation, preservation and use of records and memories of everyday life. Report GIT-GVU-02-02, Technical.

Truong, K. N., Abowd, G. D., and Brotherton, J. A. (2001). Who, what, when, where, how: Design issues of capture & access aplications. In *Proceedings of the International Conference on Ubiquitous Computing (Ubicomp 2001)*, pages 209–224, Atlanta-USA.

Weiser, M. (1991). The computer for the 21st century. *Scientific American*, 265(3):94–104.